Proceedings of the Second Nordic Workshop on Secure Computer Systems

Arto Karila, Timo Aalto (editors)

Helsinki University of Technology
Department of Computer Science and Engineering
Telecommunications Software and Multimedia Laboratory

Espoo 1997/TLM-C2

NordSec'97

SECOND NORDIC WORKSHOP ON SECURE COMPUTER SYSTEMS

"Encouraging co-operation"

**6-7 November 1997, Dipoli Conference Center,
Espoo, Finland**

ORGANIZED BY

Telecommunication Software and Multimedia Laboratory,
Helsinki University of Technology

SUPPORTED BY

Finnish Information Processing Association

SIG Security, Dataföreningen i Sverige

Nixu Oy

Data Fellows Ltd

PROGRAM COMMITTEE

Jan Ekberg, STAKES, Finland

György Endersz, Telia Research, Sweden

Erland Jonsson, Chalmers University of Technology, Sweden

Arto Karila, Helsinki University of Technology, Finland

Svein Knapskog, Norwegian University of Science and Technology, Norway

ORGANIZING COMMITTEE

Arto Karila, Helsinki University of Technology, Finland

Timo Aalto, Helsinki University of Technology, Finland

WORKSHOP PROGRAM

Thursday, November 6

08.30
Registration, Coffee
09.00
Opening, Arto Karila, Helsinki University of Technology, Finland
09.15
Invited Speaker: Tatu Ylönen, SSH Communications Security Ltd.

Session 1 - Applying theory into practice

session chair: Erland Jonsson, Chalmers University of Technology, Sweden

10.00
A Formal Task-based Privacy Model and its Implementation: An Updated Report, Dr. Simone Fischer-Hubner, University of Hamburg
10.30
Coffee break
11.00
Comparison of graph-search algorithms for authorization verification in delegation networks, Tuomas Aura, Helsinki University of Technology
11.30
A Model for Trust in Security Systems, Audun Jøsang, Norwegian University of Science and Technology, Trondheim
12.00
Lunch

Session 2 - Corporate Security

session chair: Svein Knapskog, Norwegian University of Science and Technology, Norway

13.30
Sixty Important Links in a Company's Information Security Chain, Thomas Finne, Åbo Akademi
14.00
Information Security in System Outsourcing, Tero Viiru, Kajaani Polytechnic, Jussipekka Leiwo, Monash University
14.30
IT Security Awareness - Issues for Industry, Jorma Kajava, Mikko T. Siponen, University of Oulu
15.00
Coffee break

Demonstrations: Security in Corporate Networks

15.30-17.00
Data Fellows Ltd + Nixu Oy, Jukka Kotovirta

Workshop Cocktail

19.00-
Innopoli

Friday, November 7

Session 3 - Secure Applications

session chair: Jan Ekberg, STAKES, Finland

09.00
The MobiMed Approach to Privacy in Medical Systems, Kurt Bauknecht, Ralph Holbein, Othmar Morger, Ulrich Nitsche, Stephanie teufer, Univerisity of Zurich
09.30
Fast Payment Scheme for Electronic Fee Collection, Cristian Radu, Katolieke Universiteit leuven, Rabah Boussouira, Advantec Ltd
10.00
Coffee break
10.30
Building Secure High Speed Extranets, Chandana Gamage, Jussi Leiwo, Yuliang Zheng, Monash University

Short Papers Session

session chair: Arto Karila, Helsinki University of Technology, Finland

11.00
Short Presentations
12.15
Lunch

Session 4 - Security Analysis

session chair: György Endersz, Telia Research AB, Sweden

13.45
A Taxonomy and Overview of Information Security Experiments, E. Jonsson, Chalmers University of Technology, L. J. Janczewski, The University of Auckland
14.15
A Preliminary Evaluation of the Security of a Non-Distributed Version of Windows NT, Hans Hedbom, Chalmers University of Technology, Stefan Lindskog, University of Karlstad, Erland Jonsson, Chalmers Univeristy of Technology
14.45
Coffee break
15.15
Security flaws in popular security software: Lessons learned from problems in SSH-1.2.17 and older, Antti Huima, Helsinki Univerisity of Technology
15.45
An Approach to UNIX Security Logging, Stefan Axelsson, Ulf Gustafson, Ulf Lindqvist, Erland Jonsson, Chalmers University of Technology

Conclusions

16.15-16.30
Concluding words

A Formal Task-based Privacy Model and its Implementation: An Updated Report

Simone Fischer-Hübner
University of Hamburg, Faculty for Informatics
Vogt-Kölln-Str.30, D-22527 Hamburg
Email: fischer@rz.informatik.uni-hamburg.d400.de

Abstract:
Privacy technologies are becoming more relevant, because individual privacy is at risk in the Global Information Society. In this paper, an updated version of a formal task-based privacy-model, which can be used to technically enforce legal privacy requirements, is presented. It is shown, how this model is specified and implemented according to the General Framework for Access Control (GFAC)- approach .

1. Introduction

On the way to the Global Information Society, individual privacy is at risk (see [Fischer-Hübner 1997]). The EU-directive on data protection as well as privacy laws of many western states require basic privacy principles to be guaranteed when personal data are collected or processed, such as:

- *purpose binding* (personal data obtained for one purpose should not be used for another purpose without informed consent)
- *necessity of data collection and processing* (the collection and processing of personal data shall only be allowed, if it is necessary for tasks falling within the responsibility of the data processing agency).

In the Global Information Society, privacy cannot be efficiently implemented solely by legislative means. Data protection commissioners are therefore demanding that legal privacy requirements should be technically enforced and should be a design criteria for information systems. A recent study by the Dutch Data Protection Authority and the Information and Privacy Commissioner for the Province of Ontario / Canada [Registratiekamer et al. 1995] is exploring privacy enhancing technologies that are providing anonymity or pseudonymity for the users. Also the Privacy Class of the Common Criteria [Common Criteria 1996] is focused only on anonymity, pseudonymity, unlinkability and unobservability of users.

However, besides the privacy protection of the users, there is also a need for privacy enhancing technologies to protect the data subjects (the so-called usees). Unfortunately, today's security models are mostly not appropriate to enforce basic privacy requirements, such as necessity of data processing or purpose binding. In this paper, an updated version of a formal task-based privacy model that was introduced in [Fischer-Hübner 1994], [Fischer-Hübner 1995] shall be presented, which can be used to technically enforce legal privacy requirements in a system. Examples how to apply the model to clinical information systems will be given. Besides, it will be shown, how this model has been implemented together with other security models according to the Generalized Framework for Access Control (GFAC-Approach, see [LaPadula 1995]).

2. Technical Enforcement of Privacy Requirements

Having been funded by the U.S. Government, research and development of secure systems have been mainly concentrated on maintaining secrecy of classified information. Systems have been preferably developed to enforce secrecy policies such as the one of the Bell LaPadula model, implemented by Mandatory Access Control (MAC) and Discretionary Access Control (DAC) as required by the classes of the Orange Book [TCSEC 1985] and by the example functionality classes of ITSEC [ITSEC 1991].

MAC (as defined by the Orange Book) restricts the access to objects based on the sensitivity of the information contained in the objects and the clearance of the subjects. However, personal data cannot be classified accurately by its sensitivity *per se*, because the sensitivity of personal data is related to the purpose and context of its use. As the *German Constitutional Court* proclaimed in its census decree, there are no non-sensitive personal data, because dependent on the purpose and context of use all kinds of personal data can become sensitive. There are personal data that *per se* already contain sensitive information (e.g. medical data), but dependent on the purpose and context of use, such sensitive data can become even more sensitive and data that seem to be non-sensitive (e.g. addresses) can become highly sensitive as well. In order to enforce privacy, it should be checked whether the purpose of the task, currently performed by the user who wants to access personal data, corresponds to the purpose for which that personal data were obtained (**requirement of purpose binding**).

DAC restricts access to objects based on the identity of subjects and/or subject groups. DAC permits the granting and revoking of access privileges to data to be left to the discretion of a user with a certain access permission that has control over the data. However, under privacy aspects, personal data about a data subject should not be "owned" or "controlled" by another person. In order to protect privacy, an access control decision should not be determined by the user´s identity, but by the task that the individual user is currently performing. Personal data should only be accessible to a user, if such access is necessary to perform his/her current task and if he/she is authorised to perform this task (**requirement of necessity of data processing**).

More recent security models, such as the Clark Wilson model [Clark/Wilson 1987] or Role-based Access Control models (e.g. [Ferraiolo/Kuhn 1992]) can enforce security aspects and requirements (e.g. integrity of personal data, users/roles are granted only such access rights as needed to perform their duties) that are more appropriate for the privacy protection of personal data. However, they are not especially designed to directly enforce privacy principles such as purpose binding.

3. A Formal Privacy Model

3.1 Model Definition:

In this section, the concept of a formal security model that directly enforces basic legal privacy requirements, such as *purpose binding* or *necessity of data processing*, is described

The **privacy policy** that the model shall enforce can be described informally as follows:
A user shall only have access to personal data, if this access is necessary to perform his/her current task and only, if the user is authorised to perform this task. The user shall only access

data in a controlled manner by performing a (certified) transformation procedure, for which the user's current task is authorised. Besides, the purpose of his/her current task must correspond to the purposes for which the personal data were obtained or there has to be consent by the data subjects.

This **formal task-based privacy model** is defined as a state machine model. A state machine model describes a system as an abstract mathematical state machine, where state variables represent the states of the machine and transition functions describe how the variables change. The privacy model contains the following *state variables*, *invariants, constraints (privacy properties)* and *state transition functions (model rules):*

a.) **State variables**
First, the security-relevant (or better: privacy-relevant) state variables shall be defined that are needed to formally define the privacy policy and the system states.

Subjects S: Subjects are the active entities of the system (e.g. users, processes). **S = set of current subjects = {S_1, S_2,...}.**

Active subject subj: identity of the subject that is currently active and is invoking a transition functions.

Objects O: Objects are passive entities (e.g. files, records) containing personal data. **O = set of current objects containing personal data = {O_1, O_2,....}.** Personal data are data about an identified or identifiable person.

Object-classes O-class: An object containing personal data can normally be classified by a certain class, e.g. patient record in a hospital, accounting data. **O-class = set of the different object-classes = {o-class$_1$, o-class$_2$,...}**

Object-class function Class: Each object is classified by an object-class. A function **Class: O -> O-class** is defined, where Class(O_i) is the object-class of the object O_i.

Tasks T: A subject shall be allowed to access an object only by performing a task. The tasks have to be defined for each application. **T = set of tasks = {T_1, T_2,....}.**

Current Task CT: The task that is currently performed by a subject is called its current task. If a subject is not currently performing a certain task, its current task is defined by the value Nil. A function **CT: S -> T** $\cup$ **{Nil}** is defined, where CT(S_i) is the current task of subject S_i.

Authorised Tasks AT: AT is a function that defines a non-empty set of tasks for a subject that this subject is authorised to perform. **AT: S -> $2^{T \cup \{Nil\}} \setminus \varnothing$**, where AT($S_i$) is the set of tasks that S_i is authorised to perform. $\forall$ $S_i \in$ S: Nil $\in$ AT(S_i). For an easier administration, authorised tasks could alternatively be defined for user roles instead of individual users.

User Role: Subjects can be categorised by certain security-relevant roles, which users can play. **Role: S -> user-role,** where **user-role = {user, sec-officer, data-protection-officer, tp-manager,...}**. Further user roles have to be defined, if authorised tasks are defined for user roles and not for individual users.

Users, who are defined as responsible for a tasks, shall be allowed to delegate this task. Therefore, a function "responsible" is defined: **Responsible: T -> 2^S,** where Responsible(T_i) is the set of subjects, who can request to delegate the task T_i.

Purposes P: Every task has to serve a certain purpose. Besides, personal data have to be collected for certain purposes. Purposes have to be defined according to the system´s applications. Defined purposes aremodelled by a set P of purposes: **P = set of purposes = {p_1, p_2,...}.**

The elements of T have to be defined according to the purposes of P. Each task of T has to serve exactly one purpose, but each purpose can be achieved by the performance of different tasks. Different purposes are achieved by disjunctive sets of tasks.

Purpose function for tasks T-Purpose: Each task has to serve exactly one purpose. A function **T-Purpose: T -> P** is defined, where T-Purpose(T_i) is the purpose of task T_i.

Purpose function for object class O-Purpose: Each object-class has to have specified purposes for which the personal data of this class are collected. A function **O-Purpose: O-class -> $2^P \setminus \emptyset$** is defined, where O-Purpose(O-$class_i$) are the purposes for which the object class O-$class_i$ was obtained.

Transformation Procedures TRANS: A subject cannot access an object arbitrarily. If it performs a task, it may execute certain transformation procedures, that are accessing objects in a controlled manner. **TRANS = {$transp_1$, $transp_2$,...}.**

Current Transformation Procedure CTP: The transformation procedure that is currently executed by a subject is called its current transformation procedure. If a subject is currently not executing a transformation procedure, its current transformation procedure is defined by the value Nil. **CTP: S -> Trans $\cup$ {Nil}.**

Authorised Transformation Procedures ATP: While performing a task, a subject is authorised to run certain transformation procedures. **ATP: T -> $2^{Trans \cup \{Nil\}} \setminus \emptyset$**, $\forall\ T_i \in T$: Nil $\in$ ATP(T_i).

Access rights A: The access rights that a subject can have to a personal data object are defined by access attributes **A = {read, write, append, delete, create}**.

Necessary accesses NA: For any task, it has to be defined in advance which accesses to which object-classes by running which transformation procedure are needed to perform this task. This is done by defining the set **NA** which consists of triples of the form **(T_i, o-$class_j$, $transp_k$, x).** (T_i, o-$class_j$, $transp_k$, x) $\in$ NA means that for the performance of task T_i the x-access to objects of the object-class o-$class_j$ by running transformation procedure $transp_k$ ($\neq$ Nil) is necessary, x $\in$ A.

Current access set CA: A current x-access, where x $\in$ {read, write, append}, by a subject S_i to an object O_j in the current state is represented by a triple **(S_i, O_j, x)**. The current access set CA is a set of triples representing all current accesses.

Consent C: According to most national privacy laws, the processing and use of personal data shall also be admissible, if the data subject has consented. A set **C** is defined as a set of tuples (p_i, O_j). The tuple (p_i, O_j) means that the data subjects have consented that their personal data contained in the object O_j are processed for the purpose p_i.

b.) **Invariants (privacy properties):**
The following invariants define conditions for a system state to meet specific privacy principles (a so-called privacy-oriented state). They formally define the privacy policy stated above. To enforce this privacy policy, it has to be guaranteed that these invariants are fulfilled in each system state defined by the state variables.

1. ***A subject´s current task has to be authorised for the subject (task authorisation property):*** $\forall S_i: S : CT(S_i) \in AT(S_i)$.

2. A subject´s current transformation procedure has to be authorised for the subject´s current task (TP authorisation property): $\forall S_i: S : CTP(S_i) \in ATP(CT(S_i))$.

3. ***A subject shall only have current access to an object, if the access by executing a transformation procedure to objects of this class is needed to perform the current task (necessity of data processing):***
$\forall S_i:S, O_j:O: (S_i,O_j, x) \in CA \Rightarrow (CT(S_i), Class(O_j), CTP(S_i), x) \in NA.$

4. ***A subject shall only have current access to an object, if the purpose of its current task corresponds to the purpose for that objects of this class are obtained, or if there is a consent from the data subjects (purpose binding):***
$\forall S_i:S, O_j:O: (S_i,O_j, x) \in CA \Rightarrow$
$T\text{-}Purpose(CT(S_i)) \in O\text{-}Purpose(Class(O_j)) \vee (T\text{-}Purpose(CT(S_i)),O_j) \in C$

If authorised tasks are defined for user roles, $AT(S_i)$ in the first privacy invariant has to be replaced by $AT(role(S_i))$.

c.) **Constraints:**
The privacy principles of necessity of data processing and purpose binding shall also be enforced, if personal data are created or deleted. These principles can be formulated by adding constraints, which are properties of sequences of states. An invariant defines relationships between variables within individual states. A constraint differs from an invariant, because it takes into account the relationships between values in two successive states - before and after each state transition function. In the following notation the convention of placing the symbol * behind a state variable is used to refer to the new state. The sign "=" in a function should be read as a statement of mathematical equality.

1. A subject may create an object, only if it is necessary for its current task:
$(CT(subj), o\text{-}class_k, CTP(subj), create) \notin NA \wedge O_j \notin O$
$\Rightarrow O_j \notin O^* \vee class^*(O_j) \neq o\text{-}class_k$

2. A subject may delete an object, only if it is necessary for its current task:
$(CT(subj), Class(O_j), CTP(subj), delete) \notin NA \wedge O_j \in O \Rightarrow O_j \in O^*$

3. A subject may create an object, only if the purpose of its current task corresponds to the purpose of the object´s class o-class$_k$ (purpose binding):

T-Purpose (CT (S_i)) $\notin$ O-Purpose (o-class$_k$) $\wedge$ $O_j \notin O$

$\Rightarrow O_j \notin O^* \vee$ class*(O_j) $\neq$ o-class$_k$

4. A subject may delete an object, only if the purpose of its current task corresponds to the purpose of the object´s class, or if there is a consent from the data subjects (purpose binding):

T-Purpose (CT (S_i)) $\notin$ O-Purpose (Class(O_j)) $\wedge$ (T-Purpose(CT(S_i)),O_j) $\notin$ C

$\wedge O_j \in O \Rightarrow O_j \in O^*$.

A "privacy-oriented state" is defined as a system state that satisfies the four privacy invariants1.-4. A state sequence is defined as a "privacy-oriented state sequence", if each state of the state sequence is privacy-oriented and all successive states of the sequence satisfy the four privacy-constraints 1.-4.

d.) **State transition functions:**

State transition functions that describe changes of state variables that may take place, are defined for actions such as *get access, release access, create object, delete object, change current task, execute transformation procedure, exit transformation procedure*.

For all transition functions it has been formally proven that they preserve the privacy invariants and constraints. By mathematical induction it can therefore be shown that, if a system that enforces the privacy model, starts in a privacy oriented state, then all the state sequences of the system will be privacy-oriented.

The following transition function, for example, describes how a subject can get a current access to an object:

get-access *(S_i, O_j, x)*

(Semantics: Subject S_i requests that access to object O_j in usage mode x be enabled, x $\in$ {read, write, append})

If (CT (S_i), Class (O_j), CTP(S_i), x) $\in$ NA

and

[((T-Purpose (CT (S_i)) $\in$ O-Purpose (Class (O_j))

or

(T-Purpose (CT (S_i)), O_j) $\in$ C)]

then

CA* = CA $\cup$ {(S_i, O_j, x)}

Furthermore, privileged transition functions are needed to define and change tasks, purposes, authorised tasks for a subject, authorised transformation procedures for a task, object classes and their purposes, necessary accesses and consents. These privileged functions shall be executed by the security officer. According to the 4-eyes principle, the definitions of these sets and functions should be done in co-operation with another person, who cares for the privacy interests of the data subjects (e.g. data protection officer or works council). In order to define

and implement such a joint action scheme, another system-variable for a "one-time" ticket is introduced:

A Ticket **TKT_j,(function-type, parameter-list):** is issued by a Subject S_i (who is either a user in the role "data-protection-officer" or a user, who wants to delegate a task, for which s/he is responsible) and sent to the security officer (user in the role "sec-officer"). It means that S_i requests the security officer to perform a certain function with certain parameters. TKT is defined as the set of all issued tickets. A ticket is created by performing the following privileged function:

create-ticket (S_i, function-type, parameter-list)
(Semantics: S_i requests to create a ticket TKT_i(function-type, parameter-list))

If role (S_i) = data-protection-officer $\vee$
(function-type = add-auth-Task $\wedge$ parameter-list = (S_m, T_n) $\wedge$
($S_i \in$ responsible (T_n))
then
$TKT^* = TKT \cup \{TKT_i$ (function-type, parameter-list)$\}$

With an appropriate ticket the security officer is enabled to execute a corresponding privileged function:

Data Protection Officer S_j issues ticket:

Ticket TKT_j issued by S_j
Function-type: add-NA Parameter-list: T_k, o-class$_m$, transp$_n$, x

Security Officer S_i can use this ticket to perform a privileged function:

add-NA *(S_i, TKT_j (function, T_k, o-class$_m$,transp$_n$,x)*
If role (S_i) = sec-officer
and
function-type = add-NA
and
role (S_j) = data-protection-officer
then
$NA^* = NA \cup \{ (T_k$, o-class$_m$,transp$_n$,x)$\}$
$TKT^* = TKT - \{TKT_j$ (function, T_k, o-class$_m$, transp$_n$, x)$\}$.

Figure 1: Joint action scheme for the execution of privileged functions.

3.2 Application Example:

The following example shall demonstrate how the privacy model can be applied in the environment of a university hospital.

The organisation of an university hospital is typically divided in the areas medical treatment, care, research and administration. Consequently, appropriate purposes for separating the main areas in a hospital could be:

P: MT (medical treatment), AD (administration), CAR (care) and RE (research).

Within each area, different tasks (with a task-purpose that is corresponding to the area) can be defined. Examples of possible tasks (with T-purposes in parenthesises) are:

diagnosing (MT), operation (MT), therapy (MT), intensive care (CAR), patient-admission (AD), accounting (AD), statistical analysis (RE),....

Examples for possible object classes of personal patient data are (with O-Purposes in parenthesises) : admission data (AD, MT, CAR), billing data (AD, MT), accounting data (AD), diagnosis (MT, CAR), treatment data, e.g. operation data (MT), treatment request (MT, CAR), statistics (RE),.....

Transformation procedures (TRANS) could be: accounting procedures, statistical procedures, editors, append-only-editors, type-on-screen-program, create-personal-data program, archive program,.....

Necessary Accesses (NA) could be defined as follows:

(diagnosing, diagnosis, editor, read/write/create/append)
(diagnosing, billing-data, append, append-only-editor)
(diagnosing, treatment-request, create-personal-data program, create)

(statistical analysis, diagnosis, statistical procedures, read)

Furthermore, users (or user roles) have to be authorised for the adequate tasks:
For example, a surgeon is authorised for the operation task (among others), an internist is authorised for diagnosis and therapy, a therapist is authorised for therapy, the registration staff is authorised for patient-admission, the billing staff is authorised for accounting, a researcher is authorised for statistical analysis,...

This example demonstrates the differences of the concept of necessity of data processing and the concept of purpose binding: For a researcher it might be necessary to do a statistical analysis by running a statistical program with read-access on diagnosis data. On the other hand, diagnosis data is collected exclusively for the purpose of medical treatment and the purpose of the task statistical analysis is research. Consequently, the principle of purpose binding allows an access to diagnosis data of a patient for research purposes only, if the patient has consented to it.

3.3 Information flow Control:

In the privacy model, a subject may access an object, if the purpose of his/her current task is contained in the set of purposes for which data of the object-class are obtained.
The following attack-scenario (in a hospital environment) may allow illegal information flow:

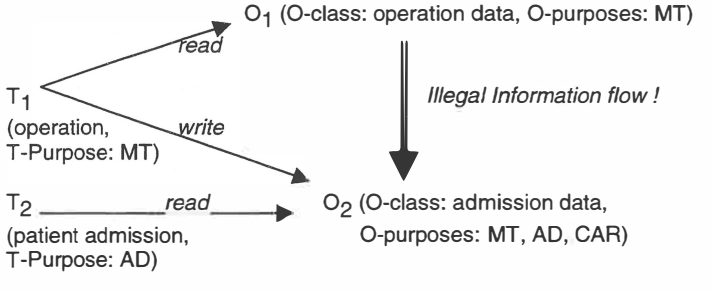


Figure 2: Illegal Information Flow

A subject performing task T_1 could read object O_1 and write sensitive data from O_1 to object O_2. The consequence is that another subject that is performing T_2 (with T-Purpose AD) could then read data from O_1 (with O-Purpose(class(O1) = {MT}), which was written to O_2. This could violate the principle of purpose binding !

Such illegal information flow can be prevented, if the following condition is guaranteed:
In any state, if a subject S_i has simultaneous read-access to object O_1 and write- or append-access to object O_2, then O-Purpose (class (O_1)) $\supseteq$ O-Purpose (class (O_2)).

Information flow control mechanisms by access control or by program certification could be used to satisfy this condition:

Certification mechanisms:
A subject can access an object only by executing a certified transformation procedure that is authorised for its current task. A program certification mechanism could check that each statement in the transformation procedure, if executed, would not cause an information flow violation. Certification mechanisms could be integrated into a compiler or flow proofs could be combined with correctness proofs to achieve a more precise certification mechanism.

Flow-secure Access Control:
A simple flow control can be integrated into the access control mechanism of the operating system. To formally model the access control of information flow, the privacy model is extended. A new state variable is introduced:

Input-Purposes: For each subject the intersection of O-purpose sets of the object classes of the objects to which the subject has had read-access, is recorded.
A function **Input-Purposes: S -> 2^P** is defined, were Input-Purposes(S_i) is the intersection set of purposes of object classes of data that S_i has read. It defines the set of purposes for that all the data, which S_i has read, were obtained. Initially (after process creation) Input-Purpose (S_i) is set to P.
Illegal information flow can be prevented, if it is guaranteed that S_i may not write to an object, which was obtained for purposes not contained in Input-Purposes (S_i). Therefore, the following security invariant and constraint have to be guaranteed as well:

Information flow control invariant:
$\forall$ S_i: S, O_j:O, x $\in$ {write, append} : (S_i, O_j, x) $\in$ CA =>
O-Purpose (class (O_j)) $\subseteq$ Input-Purposes (S_i).

Information flow control constraint:
$\forall$ S_i: S, O_j:O: (S_i, O_j, read) $\in$ CA =>
Input-Purposes* (S_i) = Input-Purpose (S_i) $\cap$ O-Purpose (class (O_j))

In order to guarantee the information flow invariant and constraint, the state transition functions have to be extended appropriately.

In this project, the privacy model has been extended with the option of flow-secure Access Control. However, the flow-secure Access Control has not been integrated in the system specification and implementation so far. First of all, illegal information flow should be prevented by a careful design of (certified) transformation procedures as well as by an appropriate definition of necessary accesses.

5. Specification and Implementation according to the Generalized Framework for Access Control Approach

In a project, it has been specified, how the privacy model can be enforced according to the Generalized Framework for Access Control Approach (GFAC) in Unix System V. The GFAC (see [Abrams et al. 1990], [LaPadula 1995]) is a framework for expressing and integrating multiple policy components to make it feasible to configure a system with security policies chosen from a vendor provided set of options with confidence that the resulting system´s security policies will be properly enforced. A draft specification how the GFAC approach enforcing the Bell LaPadula model and the Clark Wilson model can be implemented in Unix System V, published in [LaPadula 1995], was further elaborated and extended with the policy rules of the privacy model. This specification was used and adapted for the implementation and integration of the privacy model according to the GFAC-approach together with other security models (Bell LaPadula amongst others) in the Linux operating system. Linux was chosen as a demonstration system, because it is a robust system, and its source code is available. The results of the project can be easily transferred to more secure Unix versions.
The GFAC approach was chosen, because it makes it easily possible to combine the privacy model that enforces general privacy rules of national privacy legislation together with other more specific privacy regulations (e.g., privacy provisions of a hospital information law), which can be granted a higher priority.

5.1 The GFAC Approach

According to the GFAC approach, the Trusted Computer Base (TCB) consists of an access enforcement facility (AEF) and an access decision facility (ADF).

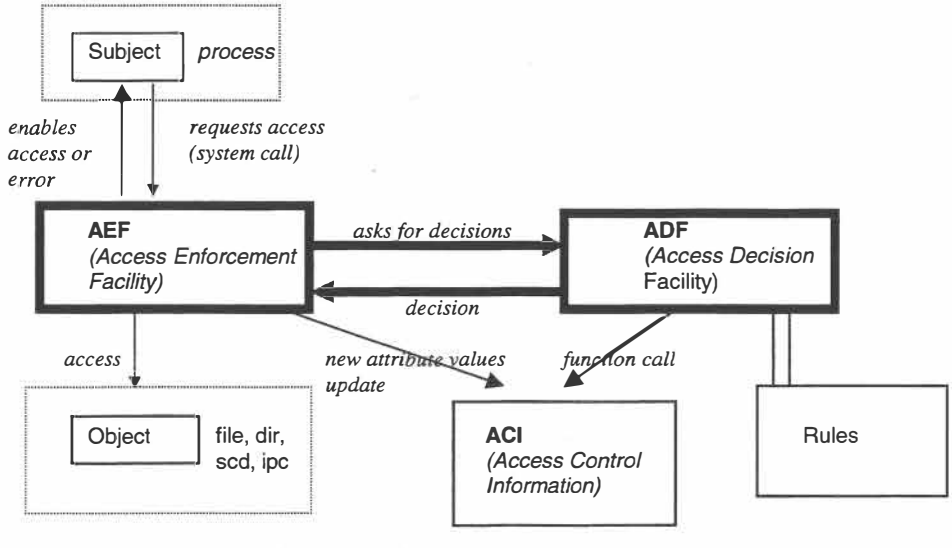


Figure 3: GFAC-Approach (see [LaPadula 1995])

ADF enforces the system´s security policies and a metapoliccy to decide whether processes´ requests satisfy those security policies. AEF uses the ADF-decisions to implement the access operations.
In order to enforce GFAC, the kernel of a system is splited into two parts: the AEF-kernel and the ADF-kernel. For each security-relevant system call, e.g. if a process requests to access an object (file, directory, security control data (scd) or interprocess communication data (ipc)), or if a process wants to clone itself or to send a signal, the AEF-kernel sends a message to the ADF-kernel to ask for access decisions. The message also references the Access Control Information (ACI, e.g. security attributes) needed by the ADF-kernel to make its decisions. The ADF-kernel evaluates its security policies by using the policy rules and Access Control Information. It then evaluates its metapolicy that uses the decisions of the different security policies to finally decide about the process´ request. The AEF-kernel then completes the system call, enabling the requested access, if the decision was favourable, or returning an error message.

5.2 The Specification and Implementation of the Privacy Model

Access Enforcement Facility (AEF):

In our implementation, because of the structure of the Linux kernel, AEF is enforced by extending all security-relevant system calls with ADF-requests and notification messages.
For each system call, there is at least one ADF-request that relates to its functionality, but also further requests might be necessary. For example, the open system call has to be extended with ADF-requests for reading a directory, creating a directory/file, truncating a file and for append-/read-/write/read-write-opening a file.
Each state transition function of the model is enforced by a system call. Therefore, AEF was extended with system calls (change-current-task, create-pers-data as well as all privileged system calls) that are exclusively needed for the privacy model.

Access Control Information (ACI):

The ACI administration and ADF are implemented as independent modules. The ACI module is responsible for a reliable administration of security attributes of processes, of users and of all resources that are needed and controlled by the security policies. Besides, it administrates other security relevant information, such as the lists of necessary accesses or of defined tasks and their security attributes. Access to ACI is only possible by defined function calls. The ACI used by ADF to make access decisions are mostly corresponding to the state variables of the privacy model.

The tables below are listing the access control information needed for the system specification and implementation of the privacy model, the corresponding variables of the privacy model and their value domains. Predefined values are printed in bold letters. ID is the abbreviation for identifier.

User-ACI	Model variable	Values
authorised-tasks	AT	a set of task-IDs including **NIL**
role	role	**sec-officer, user, data-protection_officer, tp-manager, system-admin**

Process-ACI	Model variable	Values
owner (pointer to user)		
transformation procedure	CTP	a transformation-procedure-ID or **NIL**
current_task	CT	a task-ID or **NIL**
process-type		**NIL, TP**

Object-ACI	Model variable	Values
class	class	an object-class-ID or **ipc, none**
ipc-purpose	O-Purpose *(but depending directly on the object)*	a purpose-ID or **NIL**
transformation-procedure	TRANS	a transformation-procedure-ID or **NIL**
object-type		**file, dir, ipc, scd**
data-type		**NIL, TP, personal data, non-personal-data**

Object-class-ACI	Model variable	Values
purpose	O-Purpose	a set of purpose-IDs

Further ACI	Model variable	Values
Necessary-Accesses	NA	table with entries of the form (task-ID, class-ID, transformation-proc.-ID, access-right)
Consent	C	table with entries of the form (process-ID, object-ID)
Purpose-list	P	list of purpose-IDs
Task-list	T	list of task-IDs
Ticket	TKT	records of the form (issuer, function-type, parameter-list, timestamp)

Furthermore, for tasks and for tickets the following ACI is used:

Task-ACI	Model variable	Values
purpose	T-Purpose	a purpose-ID
authorised TP	ATP	set of transformation-procedure-IDs including **NIL**
responsible	responsible	set of user-IDs

Ticket-ACI	Values
ticket-issuer	a user-ID
function-type	**add_authorised_tasks, delete_authorised_tasks, add_task, delete_task, add_NA, delete_NA add_purpose, delete_purpose, add_object-class, delete_object_class , add_authorised-TP, delete_authorised_TP, add_consent, delete_consent, add_responsible_user, delete_responsible_user, set-role, set-object-class**
parameter-list	(depending on the function-type)
timestamp	system time value

A process and the user, who is the owner of the process, are corresponding to a subject in the privacy model. In the model, only objects containing personal data are modelled. In the system specification and implementation, also other objects (passive entities) have to be considered. Therefore, the object-ACI object-type and data-type (for objects of the object-type "file") were introduced at the specification level. Objects containing personal data are typically objects with the object-type "file" and data-type "personal data". Besides, also interprocess communication (ipc-) objects with the object-type "ipc", such as message queues, can contain personal data and have to be protected. Ipc-objects have the predefined class "ipc". In contrast to personal data files, for which their purposes are depending on their object-classes, ipc-purposes are directly defined for ipc-objects. The ipc-purpose of an ipc-object containing personal data is defined by the purpose of the task, which was performed to create the ipc-object. For ipc-objects, consents by data subjects cannot be defined, and consequently, consents are not checked for ipc-objects.
If a transformation procedure is creating a file by using the normal create-system call (instead of the newly defined system call "create-pers-data"), the created file will automatically get the data-type "personal-data" and the class "none". This shall prevent that a transformation procedure can illegally transfer personal data to a non-personal-data file. The privileged "set-object-class" system call can then be used to define the data-type of the file as "non-personal-data", or to define another class, if the data are indeed personal data.

Access Decision Facility (ADF):

The ADF module receives access requests from AEF. The access requests are evaluated by the rules of the different policies (privacy policy, Bell-LaPadua policy). A metarule is using the policy decisions to evaluate the overall access decision.
A policy rule can be specified with a Case-statement

```
SELECT CASE request
    CASE request, request,.., request
        statement block
                *
                *
    CASE request, request,.., request
        statement block
END-SELECT,
```

where requests are standing for the different access-requests that are sent by AEF (see also [LaPadula 1995]).
Figure 4., for example, specifies how the privacy policy rule of ADF is deciding about the read-open request. The specification language should be intuitively understandable to a broad audience, but is also explained in [LaPadula 1995]. In this specification, the following predicates have the following meaning:
Necessary (task, class, transp, right) <=> (task, class, transp, right) is element of the table Necessary-Accesses.
Purpose-binding(task, class) <=> Purpose(task) is element of Purpose(class).
Consent(task, object) <=> (purpose(task), object) is element of the table consent.

```
CASE read-open
    SELECT CASE target[input-argument]
        CASE file
            SELECT CASE data-type(object)
                CASE personal-data
                    IF  Necessary (current-task (process), class (object),
                        transformation-procedure (process), read )
                    AND
                    Purpose-binding (current-task (process),
                                                class (object))
                    OR
                    CONSENT (current-task (process), object)
                    THEN
                            return(yes)
                    ELSE
                            return(no)
                CASE TP
                        return (no);
                CASE ELSE
                        return (do-not-care);
        CASE ipc
            IF current-task (process) is NOT NIL
            THEN
                ⌈IF Necessary (current-task(process), ipc,
                  transformation-procedure (process), delete)
                    AND
                   Purpose-binding (current-task (process),
                                            ipc-purpose (object))
                 THEN
                        return (yes)
                 ELSE
                        return (no)⌋
            ELSE
                ⌈IF ipc-purpose(object) is NOT NIL
                THEN
                        return (no)
                ELSE
                        return (do-not-care)⌋
        CASE dir
            return(do-not-care)
        CASE ELSE
            return(undefined);
    END-SELECT
```

Figure 4: Specification of the ADF- privacy policy rule decision on the read-open request

6. Final Remarks:

This project demonstrates how legal privacy requirements can be technically enforced in a system. The GFAC-approach allows to combine the privacy model with other security models as well as with models protecting more specific privacy requirements of certain application areas. In future, it is planned to test the system concept for real application scenarios.

Acknowledgement:

I want to thank Amon Ott, who was a helpful partner in the project and mainly responsible for the system implementation part.

Literature:

[Abrams et al. 1990] Marshall Abrams, K.Eggers, L.LaPadula, I.Olson, "A Generalized Framework for Access Control: An Informal Description", *Proceedings of the 13th National Computer Security Conference*, Washington, October 1990.

[Clark/ Wilson 1987] David Clark, David Wilson, "A Comparison of Commercial and Military Computer Security Policies", *Proceedings of the IEEE Computer Society Symposium on Security and Privacy,* Oakland, 1987.

[Common Criteria 1996] Common Criteria Editorial Board: Common Criteria for Information Technology Security Evaluation, Version 1.0, January 1996.

[Ferraiolo/Kuhn 1992] D. Ferraiolo, R.Kuhn, "Role-Based Access Controls", *Proceedings of the 15th National Computer Security Conference*, Baltimore MD, October 1992.

[Fischer-Hübner 1994] Simone Fischer-Hübner, "Towards a Privacy-Friendly Design and Usage of IT-Security Mechanisms", *Proceedings of the 17th National Computer Security Conference*, Baltimore MD, October 1994.

[Fischer-Hübner 1995] Simone Fischer-Hübner, "Considering Privacy as a Security-Aspect: A Formal Privacy-Model", DASY-Papers No. 5/95, Institute of Computer and System Sciences, Copenhagen Business School, 1995.

[Fischer-Hübner 1997] Simone Fischer-Hübner, "Privacy at Risk in the Global Information Society", in: Jacques Berleur and Diane Whitehouse, Eds., 'An ethical global information society: Culture and democracy revisited', *Proceedings of the IFIP-WG9.2/9.5 Corfu International Conference*, May 8-10, 1997, Chapman &Hall, 1997.

[ITSEC 1991] Information Technology Security Evaluation Criteria (ITSEC), Provisional Harmonised Criteria, June 1991.

[LaPadula 1995] Leonard LaPadula, "Rule-Set Modelling of Trusted Computer System", Essay 9 in: M.Abrams, S.Jajodia, H. Podell, "Information Security - An integrated Collection of Essays", IEEE Computer Society Press, 1995.

[Registratiekamer et al. 1995] Registratiekamer, The Netherlands & Information and Privacy Commissioner /Ontario, Canada: "Privacy-Enhancing Technologies: The Path to Anonymity", Vol.1, August 1995.

[TCSEC 1985] DoD Trusted Computer Systems Evaluation Criteria, DoD 5200.28-STD, Washington D.C., Department of Defence, 1985.

Comparison of graph-search algorithms for authorization verification in delegation networks

Tuomas Aura
Helsinki University of Technology, Digital Systems laboratory
FIN-02015 HUT, Finland; Tuomas.Aura@hut.fi

Abstract

We describe and compare several algorithms for authorization decisions from a database of certificates. The algorithms are based on well-known graph-search techniques that we enhance to handle joint-delegation certificates. Experiments on generated certificate data were done to compare the efficiency of the algorithms.

1 Introduction

Emerging key-based access control mechanisms are moving the access control decisions from centralized trusted servers to the local ones. They also shift the focus from identity of entities to authorization, the right to perform operations. The access right can be delegated anonymously from key to key with a chain of certificates. Three key-based access control systems have received wide attention: SPKI certificates [2], SDSI public key infrastructure [3], and PolicyMaker local security policy database [1]. The authorization decision in all of these systems is to some extent dependent on the chaining of signed certificates that grant authority to perform operations. Thus, they must have some mechanism for determining if a set of certificates contains a chain that authorizes an operation.

In this paper, we present and compare algorithms for authorization decisions from sets of certificates. We take a simplified view of certificates: A certificate is issued by a key. With the certificate, the issuer authorizes another key, the subject of the certificate, or a group of subject keys together, to perform an operation and to delegate this right to other keys. We view the certificates as arcs of a generalized directed graph, delegation network. Our algorithms are based on ideas from graph search.

We believe efficient algorithms of this kind to be necessary when the distributed access control systems are implemented. The tasks of making authorization decisions and maintaining the certificates is likely to be given to a specialized servers that must have predictable performance. Also, if common-place applications such as databases and file and document servers start using the new access control systems, the volume of the certificate data may become so high that the techniques for its processing should be carefully selected. Experimental results with certificate data generated according to our understanding of a typical delegation network structure show that the authorization decisions can be made efficiently from large and variable sets of certificates as long as some care is taken in designing the algorithms for the purpose.

We begin by introducing our abstract view of the delegation networks and the authorization problem in Sec. 2. Sec. 3 describes the algorithms. In Sec. 4 we give results of experiments and comparisons between the algorithms. Sec. 5 concludes the paper.

2 Delegation network

We first define the terminology and authorization problem and then proceed to describe what the typical delegation networks look like.

2.1 Definitions

In our discussion, a *certificate* is a four-tuple $< issuer, subjects, k, authorization >$. *issuer* is the key signing of the certificate and *subjects* is a set of n keys to whom the certificate has been given. k is the *threshold* value determining how many subjects must co-operate to use and further delegate the right specified by *authorization*.

A *delegation network* is a set of certificates. The issuers and subjects in the certificates are called *keys*. A delegation network can be stored as a (generalized) directed graph structure where the certificates serve as directed arcs and keys as nodes. The joint-delegation certificates can be viewed as generalized arcs. If all certificates have only a single subject, the delegation network becomes a standard directed graph.

In our simplified model, delegation is always transitive. (In actual systems, the transitivity can be limited. In SPKI, for example, further delegation can be forbidden in the certificate. The algorithms in this paper are easily adaptable to such limitations.)

The authorization problem can be formulated as the existence of a computation [2]. Instead, we will try to help the reader visualize the problem with the help of trees. This is important because in order to understand the algorithms in this paper, the delegation network should be visualized as a graph. We also assume that the certificate database is set up as a graph so that the keys and their certificates are easily accessible from one another.

The tree-based definition goes as follows: A delegation network authorizes a client key c to perform an operation for a server key s if a finite tree can be formed such that

1. All nodes of the tree are marked with a key. The key on the root node of the tree is s. The key on every leaf node of the tree is c. The same key can be used for multiple nodes.

2. All nodes that have children (i.e. non-leafs) are marked with a certificate. The issuer of the certificate must be the key on that node. The subjects of the certificate must correspond to the keys on the children. If the threshold value of the certificate is lower than the number of subjects, only the threshold number of children are required. The same certificate can be used for multiple nodes. For a certificate with only a single subject, this means that there is a single child node and the key on the child node is the same as the subject. (Actually, we are not as much marking the nodes with certificates but the connection between the parent node and the group of children.)

Thus, the question "Does a delegation network authorize key c to perform operation s from server s?" can be stated as: "Does a tree exist that conforms to the above requirements?" With the graph-search algorithms, we are trying to find this kind of tree in the generalized

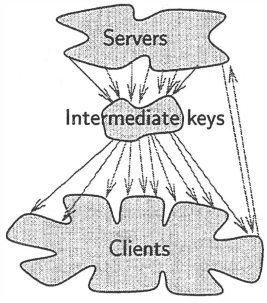


Figure 1: Typical delegation network structure

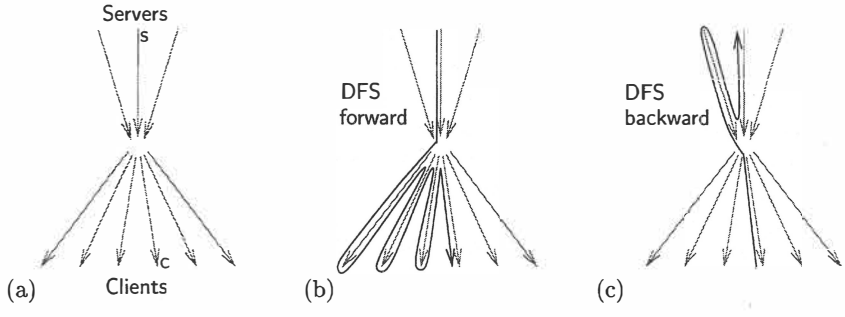


Figure 2: When forward branching is greater, backward search is faster

directed graph formed by the keys and certificates. When there are no joint-delegation certificates, the tree is reduced to a path, and standard path-finding algorithms for directed graphs can be used.

2.2 Typical delegation network structure

The delegation networks in practice, however, will not be arbitrary graphs but they will have certain structure. Although the system architectures themselves do not constrain the associations between the keys, common practices will arise from the way popular applications choose to chain their certificates.

We anticipate that most delegation networks will have an hourglass shape (Fig. 1). On the top of the hourglass there are the servers and on bottom the the clients. Direct certificates between the servers and clients are scarce. Instead, the access rights are distributed to the clients by a network of intermediate keys. These can be trusted certificate databases near the clients, reference monitors near the servers, and service brokers between them. In the extreme case, there could be a single broker delivering access rights from servers to clients, as in Fig. 2(a).

Application programs, user platforms and specialized servers themselves are unlikely to incorporate wide capabilities for maintaining valid certificates. Therefore, trusted servers are needed for certificate acquisition, updates, bookkeeping and verification. These servers will

need algorithms for authorization decision from large sets of certificates, and they themselves form an additional key layer in the network.

Naturally, the common structure will only hold for majority of the certificates. There may be occasional short links and even certificates from clients to servers. Also, the servers or clients can create a wealth of mutual relationships amongst themselves. Thus, the system must be able to accommodate arbitrary certificates between arbitrary keys. Nevertheless, we will optimize the efficiency of our algorithms to with the hourglass structure in mind.

3 Algorithms for access control decision

In the literature, no actual algorithms for authorization decisions have been described. The SPKI document [2] explicitly states that its authors believe that the authorization questions can be answered but no implementation exists for the time being. In this section, we will shortly refer to the semantical definition of the SPKI certificates and then describe several algorithms for the authorization decisions.

Two things are worth noting about our algorithms. Firstly, they are based on simple path-finding algorithms for directed graphs. We have not considered any pre-computation techniques. Storing the precomputed results or some partial information in the memory can lead to constant-time algorithms but the memory space required is $O(n^2)$ with respect to the size of the certificate database. This does not seem feasible for the implementations that we have in mind, although some kind of caching might improve the efficiency of our algorithms. Secondly, signature verification is not part of the algorithm. All signatures are verified at the time when the certificates are entered into the database.

3.1 Five-tuple reduction

The SPKI document defines the semantics of the certificates with a five-tuple reduction. (The five-tuples are certificates almost like our four-tuples.) That is, rules are given for how two certificates (or more in the case of joint delegation) reduce into one. A server should grant access to a client if there is a path of certificates that recursively reduces to one certificate where the server itself authorizes the client.

The five-tuple reduction can be used as an implementation technique. When the client asks intermediate keys to sign the reduced certificates, no signature needs to be verified more than once. Still, the client, or an entity trusted by the client, must maintain the certificates and decide when it needs to start reducing a path. Therefore, techniques are needed for efficient path finding and decision making from a set of certificates even when the five-tuple reduction is actually implemented.

3.2 Depth-first search forward

The most straight-forward way to verify authorization from a certificate graph is depth-first search in the certificate graph tracing the flow of access rights from the server to the client. The recursive search procedure has, in fact, been proposed as an alternative semantic definition of authorization for the SPKI certificates [4].

Pseudo-code for a recursive depth-first search algorithm is listed in Listing 1. The algorithm should contain no surprises to the reader. For each certificate, it counts the valid

```
 1 function dfsForward (server,client,operation)
 2     return dfsForwardRecursive(server,client,operation);
 3
 4 function dfsForwardRecursive (key,client,operation)
 5     mark key as in search path;
 6     if (key = client) return TRUE;
 7     for c in certificates signed with key
 8         if (c authorizes operation)
 9             countPaths = 0;
10             for (subj in subjects of c)
11                 if (countPaths < number of paths required by c
12                     AND (subj marked as having path to client
13                          OR (subj NOT marked as having path to client
14                              AND subj NOT marked as in search path
15                              AND dfsForwardRecursive(subj,client,operation))))
16                     countPaths = countPaths + 1;
17             if (countPaths ≥ number of paths required by c)
18                 mark key as having path to client;
19                 return TRUE;
20     unmark key as in search path;
21     return FALSE;
```

Listing 1: Depth-first search forward from server to client

paths leading from subjects of the certificate to the client. If the count reaches the threshold required by the certificate (the threshold is 1 for non-joint delegation), there is a valid authorization path from the issuer of the certificate to the client.

Unfortunately, the number of paths in a graph grows exponentially with the graph size. Fig. 3(b) shows an example of how forward search must process some nodes again even though they have been visited before. (This is a good test case for algorithmic improvements.) If all certificates had only one subject, a linear algorithm could be used instead.

Our implementation that was used for the experiments reported in Sec. 4, does several further optimizations to avoid retraversing paths. Although these significantly reduce the number of keys processed, the complexity of the algorithm remains exponential. Typically, existing certificate paths are found fast but negative answers can take even millions of steps.

3.3 Depth-first and breadth-first search backward

In Fig. 2, there is a simple hourglass shaped delegation network. Part (b) shows how a forward depth-first search from the server finds the client. In part (c), the same kind of search is initiated backward from the client to find the server. The searches are functionally equivalent (even the same algorithm can be used when all certificates have only one subject), but since the network branches less in the backward direction, the backward search is faster. That is, in a typical delegation network, backward search will perform better than forward search.

There are two possible ways for handling joint-delegation certificates in backward search.

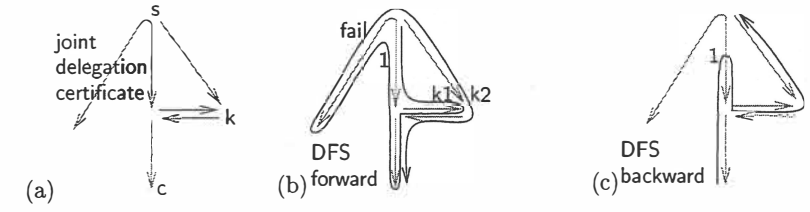


Figure 3: Depth-first forward search can visit the same node several times.

```
22 function dfsBackward (server,client,operation)
23     return dfsBackwardRecursive(client,server,client,operation);
24
25 function dfsBackwardRecursive (key,server,client,operation)
26     mark key as having path to client;
27     if (key = server) return TRUE;
28     for c in certificates given to key
29         if (c authorizes operation
30             AND issuer of c NOT marked as having path to client)
31             countPaths = 0;
32             for (subj in subjects of c)
33                 if (subj marked as having path to client)
34                     countPaths = countPaths + 1;
35             if (countPaths ≥ number of paths required by c
36                 AND dfsBackwardRecursive(issuer of c,server,client,
37                                          operation)
38                 return TRUE;
39     return FALSE;
```

Listing 2: Depth-first search backward from client to server

One can span forward searches from the subjects in order to determine immediately if enough paths from the subjects to the client exist. This approach suffers from the poor performance of the forward search. Instead, we have chosen to count the number of paths leading from the client to the subjects, and to continue the backward search from the issuer of the joint-delegation certificate when the threshold value is reached. This appears to be simple and effective. If the same subject never appears twice in the same certificate, the counting can be optimized by keeping counters with the certificates. In the pseudocode of Listing 2, we have chosen the most general approach and recount the subjects on every visit. Fig. 3 illustrates how the backward search processes every key at most once.

Backward search can also be done in breadth-first order. The breadth-first algorithm processes keys by increasing distance from the client. Like in the depth-first algorithm, the issuers of joint-delegation certificates are discarded until enough of the subjects of the certificate have been processed. In theory the breadth-first search can require more memory than the depth-first search. In our experiments, however, the memory consumption was so small that we found it difficult to give any estimates.

```
40 function bfsBackward (server,client,operation)
41     nextKeys = {client};
43     mark client as having path to client;
44     while (nextKeys = ∅)
45     currentKeys = nextKeys;
46         nextKeys = ∅;
47         for key in currentKeys
48             for c in certificates given to key
49                 if (c authorizes operation
50                     AND issuer of c NOT marked as having path to client)
51                     countPaths = 0;
52                     for subj in subjects of c
53                         if (subj marked as having path to client)
54                             countPaths = countPaths + 1;
55                     if (countPaths ≥ number of paths required by c)
56                         if (number of certificates given to issuer of c > 0)
57                             nextKeys = nextKeys ∪ {issuer of c};
58                         mark issuer of c as having path to client;
59                         if (issuer of c = server) return TRUE;
60     return FALSE;
```

Listing 3: Breadth-first search backward from client to server

3.4 Two-way search

The graph search can be optimized by starting from both ends and meeting in the middle of the path. This is illustrated in Fig. 4.

The average cost c of finding the path between two nodes in a graph grows exponentially with the length of the path d. By searching from both ends and meeting in the middle, we can reduce the problem to two parts with path length $d/2$. This way, the complexity decreases to approximately the square root of the original. In normal graphs, we can search both ways and mark visited nodes along the way. When one search finds a node visited by the other, we know that a path exists. In order to find the complete path, the search that had first visited the node and already left it must retraverse the graph to find that node again, unless memory can be used to remember the paths to all visited nodes. When we do not have excess memory at disposal, the two-way search thus reduces the cost of deciding the existence of a path between two nodes to $2\sqrt{c}$ and the cost of finding the path to $3\sqrt{c}$. (This is, of course, not complete mathematical treatment but it should give an idea of the magnitude of the expected benefits.)

When the branching factors of the graph in the two directions are different, as in our case, the efficiency is not improved quite as much but still significantly. The reason is that one-way search is always done in the direction of smaller branching factor while two-way search must also go in the less beneficial direction. If the branching factors can be estimated, the meeting point should be set nearer the end from which the branching is greater, not half-way between. If the distance d and the branching factors β_1 and β_2 are large enough, the optimal meeting point is distance

$$d \log \beta_2 / (\log \beta_1 + \log \beta_2)$$

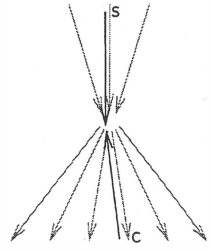


Figure 4: Two searches meet in the middle

away from the end from which the branching is β_1.

We implemented the two-way search by first doing depth-first search forward from the server, marking the visited keys on the way, and then trying to find a marked key with breadth-first (or depth-first) search from the client. The forward search algorithm is a simplified version of Listing 1 that ignores joint-delegation certificates and, thus, needs to visit every key only once. The forward search is terminated at a specified maximum depth. Experiment showed that in practice the above formula cannot be used to determine the optimal depth. Instead, a constant value of one or two should be used unless the delegation paths are especially long.

Unfortunately, the gains of two-way searching do not seem to be as big in practice as in the theoretical discussion above. The main reason is that the joint-delegation certificates make forward search from the server to the client almost infeasible. Therefore, it is not advisable to come more than one or two certificates away from the server to meet the backward search. Nevertheless, when large numbers of servers sign certificates for a few intermediate keys, the benefits of first marking keys one or two keys away from the server can be noticeable. If all certificates have only a single subject, the situation is quite different because also the forward search can be done more efficiently.

4 Experimental results

Experiments conducted with generated certificate data show that the gains from two-way search are not quite as big as expected. The backward search algorithms appear almost as efficient.

4.1 Generating certificate data

Since no real-world certificate databases are available for the time being, we generated random delegation networks with the assumed hourglass structure. This was done by dividing the keys to several levels, the top level representing servers and the bottom level clients. The number of keys on each level and the amounts of certificates between each two levels were chosen according to our (admittedly vague) idea of the typical system. The network was then automatically constructed by assigning the certificates between random keys in the specified levels.

The data presented here was collected from a network with 4 layers of keys. Table 1 shows the number of keys on each level and the certificates between them. It should be noted

Level	# keys	# certificates		from level				Number of subjects	% of certificates
				1	2	3	4		
1	100		1	5	200	10	100	1	80
2	10	to level	2	2	2	200	10	2	15
3	100		3	2	2	5	20000	3	3
4	5000		4	2	2	2	500	4	2

Table 1: Parameters for the generated delegation network

	Search algorithm			
Decision	dfs forward	dfs backward+forward	dfs backward	bfs backward
all	3273	4327	56	54
positive	3581	4210	53	51
negative	2347	4676	64	64

Table 2: Average number of algorithmic steps in for a key pair in different algorithms

that in our sample network, there are only few backward arcs towards the server. (This is determined by the lower half of the matrix giving certificate counts.) We found the results of comparisons between algorithms to be relatively stable with small changes in the parameter values. The amount of backward arcs and the arcs inside the levels, however seemed to have great effect on the efficiency of the forward depth-first search. Although we have chosen the parameters to somewhat help that algorithm, the results will not be too favorable in any case.

4.2 Results of comparison

The experiments with different one-way algorithms showed that the breadth-first backward search and depth-first backward search perform best (see Table 2). Any performance differences between these two algorithms were insignificant and certainly much smaller than differences caused by implementation details. The depth-first forward search and the depth-first backward search that spans forward searches at joint-delegation certificates, performed badly.

In the delegation network of Table 1, the forward searches took about 50 times more time than the pure backward searches. The efficiency of the depth-first search is greatly dependent on the degree of completeness of the graph and on the number of backward arch from levels near the client to levels near the server. These arcs create more paths in the graph, and the depth-first search may traverse a lot of them. The positive answers are usually returned quite fast while negative results may require exponentially more work. In some networks, the forward searches become painfully slow taking occasionally millions of steps to complete queries with negative result.

In the comparisons, the lookahead test of the breadth-first backward search (Listing 3, line 56) was disabled. We observed that the lookahead can reduce the number of keys processed in the algorithm by about up to 70 %. The more pure clients, i.e. keys that only receive certificates, there are, the more significant the speed-up will be. Hence, the optimization is in many situations more significant than it first seems.

Two-way search was tested by the first starting depth-first forward from the server to a

	Depth of forward search				
Decision	0	1	2	3	4
all	56	42	67	1517	1606
positive	51	32	58	1714	1804
negative	70	71	92	970	1053

Table 3: Average number of algorithmic steps for a key pair in two-way search

	Depth of forward search				
Decision	0	1	2	3	4
all	58	36	73	1900	1895
positive	50	21	60	2065	2048
negative	81	82	116	1370	1406

Table 4: Average cost in two-way search with no joint delegation

constant depth, and then looking for the marked nodes with breadth-first search backward from the client. (The depth-first search ignored all joint-delegation certificates; see Sec. 3.4.) Table 3 shows how the cost of computation varied in the two-way search as a function of the depth of the forward search.

The one-step forward search gave the best results. This is probably because that one step away from the client saves a lot of work in going through the large number servers attacked to a single broker. The savings amount to only 25 %. In experiments with other delegation network parameters, the best results were also given by forward search to the depth of one, or sometimes two, certificates. The savings in computation time were between 10 and 50 %. Thus, the two-way search does not perform as well as one would expect. It is also important to note that setting the maximum depth for the forward search to a number higher than 2 is risky: the complexity usually jumps up at depth 3 or 4. In delegation networks with significantly more than five distinguishable levels of keys, the desirable depth of the forward search could be higher, but we find such networks are unlikely to exits.

Table 4 shows the same kind of measurements as Table 3, only for a network without any joint-delegation certificates. Here we can see that the two-way search saves about 60 % of the cost for queries where a valid path is found. The performance improvement is much bigger than in the general delegation network. This is natural because the forward search part in the two-way search cannot handle well joint-delegation certificates.

It is also interesting to compare the last column of Table 2 with the first column of Table 3. These figures should be approximately equal. Both experiments were done by averaging the execution costs for over 1000 key pairs from the same delegation structure. The variation seems to be always greater in the searches with negative answer but we expect such queries to be minority in actual systems.

5 Conclusion

We described and compared several algorithms for authorization decisions from a database of certificates. The algorithms are based on well-known graph-search techniques that have been

enhanced to handle joint-delegation certificates. Measurements on generated certificate data were done to compare the efficiency of the algorithms.

The main observations was that it is feasible to make authorization decisions from large delegation networks comprising thousands of keys and certificates. The most efficient algorithm was found to be the two-way search where we first mark keys one or two certificates away from the server with a forward search and then try to locate one of the marked nodes with a backward search. This is a little faster than simple backward search, but more difficult to implement. The two-way search is at its best when there are no joint-delegation certificates while the backward searches handle them particularly well.

References

[1] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proc. 1996 IEEE symposium on Security and Privacy*, pages 164–173. IEEE Computer Society Press, May 1996.

[2] Carl M. Ellison, Bill Franz, Butler Lampson, Ron Rivest, Brian M Thomas, and Tatu Ylönen. Simple public key certificate. Internet draft, SPKI Working Group, July 1997.

[3] Ronald L. Rivest and Butler Lampson. SDSI — A simple distributed security infrastucture. Technical report, April 1996.

[4] Tatu Ylönen. Proposal for SPKI certificate formats and semantics. Unpublished manuscript, April 1997.

A Model for Trust in Security Systems

Audun Jøsang *

Department of Telematics, NTNU, 7034 Trondheim, Norway **
Email: ajos@item.ntnu.no

Abstract. This paper describes a formal model for reasoning about trust in information security. Trusting a system is the same a believing that it will resist malicious attacks, and trusting a human agent is the same as believing that she will cooperate. Trust therefore is a belief, and a model for trust is also a model for beliefs. The trust model presented here consists of a model for representing beliefs and a set of operations to combine beliefs. We show that this framework is suitable for modelling trust in general and give an example which illustrates how it can be applied to key authentication in PGP.

1 Introduction

Trust is a subjective belief, and not for example a property of a system or an agent. The purpose of modelling trust must therefore be to model how a human observer would asses the security of a system or the honesty of an agent. This would enable automatic appreciation of trust *as a human would do it.* The advantage would be that trust can be assessed quickly, and that situations which would have been too complex for the human mind can be efficiently analysed.

In this paper we present a flexible model for reasoning about trust in security systems. It is based on a model for representing belief in general, and since trust simply is a belief, it is also suitable for modelling trust. The model is more complete than previously proposed trust models (see e.g. [RS97, Jøs97] for an analysis of some models). The model also contains of a set of operations for manipulating these opinions and for making decisions based on opinions. An example will illustrate how the model can be applied.

2 The Trust Model

In order to better understand the philosophy behind the model, we will start with a short phenomenologic account for the concept belief, and show how this leads to a concise model for expressing trust.

2.1 Modelling Belief

We assume the world to be in a particular state at any given time. Our knowledge about the world is never perfect so we can never determine the state exactly. For the purpose of believing a proposition about the world, we assume that it is either true or false, and not something in between. Because of our imperfect knowledge, it is impossible to know with certainty whether it is true or false, so that we can only have an *opinion* about it, which translates into degrees of belief or disbelief.

In addition it is necessary to take into consideration degrees of ignorance, which can be described as a vacuous belief which fills the void in the absence of both belief and disbelief. For a single opinion about a proposition, we assume that

$$b + d + i = 1, \quad \{b, d, i\} \in [0, 1]^3 \tag{1}$$

where b, d and i designate belief, disbelief and ignorance respectively. Without going into detail, it can be mentioned that this way of representing beliefs has parallels to the Dempster-Shafer theory of evidence[Sha76]. Eq.(1) describes a triangle as illustrated in Fig.1, and an opinion about a proposition can now be uniquely described as a point $\{b, d, i\}$ in the triangle.

* This research was supported by Norwegian Research Council Grant No.116417/410.
** This research was partly carried out while the author was visiting the ISRC at QUT.

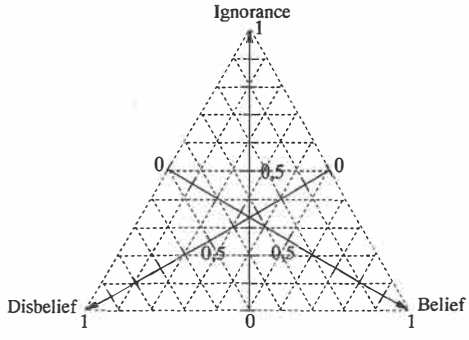


Fig. 1. Opinion Triangle

Definition 1. *Let $\pi = \{b, d, i\}$ be a triplet which satisfies Eq.(1), where the first, second and third element correspond to belief, disbelief and ignorance respectively. Then π is called an opinion. The set of opinions is denoted by* Π.

The bottom line between belief and disbelief in Fig.1 represents situations with zero ignorance and is equivalent with a traditional probability model. The degree of ignorance can be interpreted as the lack of evidence to support either belief or disbelief. The justification for Eq.(1) and for the opinion triangle is that the human mind maintains belief and disbelief simultaneously as quasi-complementary, and that when both beliefs become weaker, due to insufficient evidence, ignorance increases.

For example, the opinion {0, 0, 1} which represents total ignorance can be interpreted as the belief that it is absolutely certain that the proposition is either true or false, but there is no evidence to indicate that one is more likely than the other. The difference between {0, 0, 1} and {0.5, 0.5, 0.0} is that in the first case there is no reason to believe that the proposition is true or false, whereas in the second case there are equally strong reasons to believe both.

Opinions are considered subjective, and will therefore have an ownership assigned whenever relevant. In our notation, superscripts indicate ownership, and subscripts indicate the proposition to which the opinion apply. For example

$$\pi_p^A$$

is an opinion held by agent A about the truth of proposition p.

2.2 Modelling Trust

Imagine an observer A who is considering her trust in a particular system. She can form the proposition p: "*The system will resist malicious attacks.*" Now, her her trust in the system will be her belief in p, expressed as π_p^A.

Let the same observer A consider her trust in a particular human agent. She must assume that the agent will either cooperate or defect. She can form the proposition q: "*The agent will cooperate.*" Her trust in the agent can simply be expressed as π_q^A, which is the belief that he will cooperate.

In a similar way, trust in the authenticity of a cryptographic key can be expressed by defining r: "*The key is authentic.*" and express the opinion π_r^A.

These simple examples demonstrate that trust easily can be expressed as an opinion.

3 Second Order Probability Representation

In this section we show that opinions can be represented as 2-order probability estimates, and define a bijective mapping between the two representations. By describing the operations *consensus* and *ordering* for 2-order probability estimates, we then have the necessary formal basis for defining the equivalent operations for opinions in Sec.4.

3.1 Representation of 2-Order Probability Estimates

Let us consider the probability of throwing for example a "five" with a fair dice. This, most people would agree to be $1/6$. Imagine now a dice which has been loaded, so that the probability of throwing a particular side, without knowing which, is $1/2$, and the probability of throwing any of the other sides is $1/10$. Let us again consider the probability of throwing a "five" with this loaded dice. A 1-order probability analysis would dictate this to be $1/2 \cdot 1/6 + 1/10 \cdot 5/6 = 1/6$, again the same result as for the fair dice. But with the available information, any observer would know better; it can impossibly be $1/6$. As a matter of fact, it is either $1/2$ (with probability $1/6$) or $1/10$ (with probability $5/6$).

A 2-order probability is simply the probability of a 1-order probability. The above example illustrates a simple case of 2-order probability representation, i.e probabilities of probabilities, in which the 1-order probability could only have the two possible discrete values $1/2$ and $1/10$. In general, 2-order probabilities can be represented as a probability density function over a 1-order probability variable.

Definition 2. *Let θ be the 1-order probability variable of some binary event, then θ is defined in the interval $[0,1]$. A 2-order probability density function on θ is a function $\psi(\theta)$ so that:*

$$\begin{aligned} &\textbf{a.}\ \psi(\theta) \geq 0 \qquad \textit{for all}\ \ \theta \in [0,1], \\ &\textbf{b.}\ \int_0^1 \psi(\theta)\mathrm{d}\theta = 1 \end{aligned}$$

We will designate by Ψ the class of 2-order probability density functions, or 2-order pdfs for short.

It is natural to choose the beta function as a mathematical approximation of a 2-order pdf, since it is one of the few well known distributions that give the cumulative probability 1 to the finite interval [0,1].

The beta-family of distributions is a continuous family of functions indexed by the two parameters α and β. The beta(α, β) pdf is:

$$f(\theta \mid \alpha, \beta) = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \theta^{\alpha-1}(1-\theta)^{\beta-1}\ ,\ \ 0 \leq \theta \leq 1,\ \alpha > 0,\ \beta > 0 \tag{2}$$

As the parameters α and β vary, the beta distribution takes on many shapes. The pdf can be strictly increasing ($\alpha > 1, \beta = 1$), strictly decreasing ($\alpha = 1, \beta > 1$), U-shaped ($\alpha < 1, \beta < 1$), or unimodal ($\alpha > 1, \beta > 1$). The case $\alpha = \beta$ yields a pdf symmetric about $1/2$, and when $\alpha = \beta = 1$, the beta distribution reduces to the uniform distribution. Fig.2 illustrates some beta distributions.

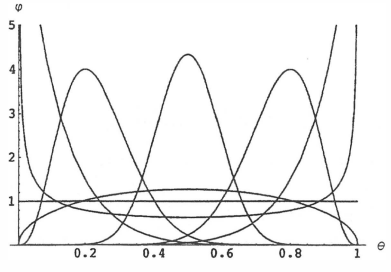


Fig. 2. Beta distribution functions

For reasons which will become clear with Th.4 and also in Sec.3.2, we will only consider the subclass of beta distributions called *2-order B-pdfs*.

Definition 3. *Let Φ be the class of beta distributions for which $\alpha \geq 1$ and $\beta \geq 1$. Let φ be a beta distribution in Φ. Then φ is called a 2-order Bayesian probability density function, or 2-order B-pdf for short.*

In our notation, 2-order B-pdfs will be characterised by the parameters $\{r, s\}$ instead of $\{\alpha, \beta\}$ through the following correspondence:

$$\alpha = r + 1, \quad r \geq 0 \quad \text{and}$$
$$\beta = s + 1, \quad s \geq 0.$$

Let φ be a 2-order B-pdf over the 1-order probability variable θ. In our notation φ will then be characterised by r and s according to:

$$\varphi(\theta \mid r, s) = \frac{\Gamma(r+s+2)}{\Gamma(r+1)\Gamma(s+1)} \theta^r (1-\theta)^s , \quad 0 \leq \theta \leq 1, \ r \geq 0, \ s \geq 0 \tag{3}$$

The choice of only considering beta distributions from Φ, which excludes all U-shaped distributions, will not cause any restrictions on the generality of our results. As a matter of fact, it can be shown that any 2-order probability distributions can be approximated as a linear combination of 2-order B-pdfs.

Theorem 4. *Let ${}_\epsilon\varphi$ be a linear combination of 2-order B-pdfs in the form of*

$${}_\epsilon\varphi = \sum_{i=1}^{n} \varphi_i \delta_i, \quad n \text{ is positive integer}, \ \delta_i \in (0, 1], \ \sum_{i=1}^{n} \delta_i = 1 \tag{4}$$

For any element $\psi \in \Psi$, ψ can be approximated with arbitrary accuracy as $\psi \approx {}_\epsilon\varphi$.

We will call ${}_\epsilon\Phi$ the set of linear combinations of 2-order B-pdfs in the form of (4).

The example with the loaded dice can in fact be represented as a linear combination of two 2-order B-pdfs in the form of peak-functions, which is graphically illustrated in Fig.3.

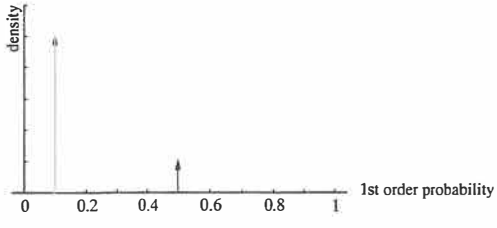


Fig. 3. 2-order probability estimate of throwing a "five" with loaded dice

3.2 Estimating the Posteriori 2-Order Probability

Assume an entity which produce events where the outcome can either be positive or negative. Let the entity produce the event a certain number of times, and let p and n designate the number of positive and negative results respectively. We assume that the priori probability estimate can be described as the uniform distribution $\varphi(\theta \mid 0, 0)$. It can then be shown that the posteriori probability estimate is a 2-order B-pdf with $\{r, s\}$ parameters equal to $\{p, n\}$.

$$\varphi(\theta \mid p, n) = \frac{\Gamma(p+n+2)}{\Gamma(p+1)\Gamma(n+1)} \theta^p (1-\theta)^n , \quad 0 \leq \theta \leq 1, \ p \geq 0, \ n \geq 0 \tag{5}$$

3.3 Consensus Between Independent 2-Order Probability Estimates

Assume two agents A and B having observed an entity produce a binary event over two different periods respectively. According to (5), their respective 2-order B-pdfs are then $\varphi(p^A, n^A)$ and $\varphi(p^B, n^B)$. Imagine now that they combine their observations to form a better estimate of the event's probability. This is equivalent to an imaginary agent $[A, B]$ having made all the observations and who therefore can form the 2-order B-pdf defined by $\varphi(p^A + p^B, n^A + p^B)$. This result can be generalised to cover real pdf parameters.

Definition 5. *Let $\varphi(r_p^A, s_p^A)$ and $\varphi(r_p^B, s_p^B)$ be two 2-order probability estimates respectively held by the agents A and B regarding the truth of a proposition p. The 2-order probability estimate $\varphi(r_p^{A,B}, s_p^{A,B})$ defined by*

$$\begin{array}{l} \textbf{1.}\ r_p^{A,B} = r_p^A + r_p^B \\ \textbf{2.}\ s_p^{A,B} = s_p^A + s_p^B \end{array}$$

is then called the Bayesian consensus rule for combining A's and B's estimates, as if it was an estimate held by an imaginary agent $[A, B]$. By using the symbol $\oplus$ to designate this operation, we get $\varphi(r_p^{A,B}, s_p^{A,B}) = \varphi(r_p^A, s_p^A) \oplus \varphi(r_p^B, s_p^B)$.

It is easy to prove that $\oplus$ is both commutative and associative which means that the order in which probability estimates are combined has no importance. Probability estimate independence must be assumed, which obviously translates into not allowing an entity's probability estimate to be counted more than once.

3.4 Ordering

Assume an agent having made 2-order probability estimates about different propositions. Sometimes we are interested in determining the "strongest" of these estimates, so that a set of 2-order probability estimates in fact can be ordered. Intuitively, the strength of a 2-order probability is its mean value which is the same as the corresponding 1-order probability estimate. However, different 2-order probability estimates can have equal mean value. In that case the estimate with the highest evidence basis (i.e. $r+s$) is the stronger.

The mean value of a 2-order probability estimate represented in the form of (3) is given by:

$$\text{Mean}(\theta) = \frac{(r+1)}{(r+s+2)}, \quad r \geq 0, \ \ s \geq 0 \tag{6}$$

By defining a Max-operator, 2-order probability estimates can be ordered.

Definition 6. *Let φ_p and φ_q be an agent's 2-order probability estimates regarding two distinct propositions p and q. Let $\varphi_{p\uparrow q}$ be the 2-order probability estimate which the following algorithm would return:*

```
Max(INPUT: φp, φq, OUTPUT: φp↑q)
BEGIN PROCEDURE
   IF φp and φq have different (r+1)/(r+s+2) ratios
   THEN
     RETURN opinion with greatest (r+1)/(r+s+2) ratio;
   ELSE
     RETURN opinion with greatest r+s;
   ENDIF;
END PROCEDURE;
```

Then $\varphi_{p\uparrow q}$ is called the maximum of φ_p and φ_q, representing the strongest 2-order probability estimate. By using the symbol "$\uparrow$" to designate this operation, we get $\varphi_{p\uparrow q} = \varphi_p \uparrow \varphi_q$.

The Max-operator $\uparrow$ can be used to order any set of 2-order B-pdfs.

3.5 Equivalence between Opinions and 2-Order B-Pdfs

We have defined Φ to be the class of 2-order B-pdfs, and Π to be the class of opinions. Let $\pi_p = \{b_p, d_p, i_p\}$ be an agent's opinion about proposition p, and let $\varphi(r_p, s_p)$ be the same agent's estimate of p being true expressed as a 2-order B-pdf. Let $\varphi(r_p, s_p)$ be defined as a function of π_p according to:

$$\begin{cases} r_p = \frac{b_p}{i_p} \\ s_p = \frac{d_p}{i_p} \end{cases} \tag{7}$$

We see for example that $\pi = \{0, 0, 1\}$ which expresses total ignorance corresponds to the uniform $\varphi(0, 0)$, that $\pi = \{1, 0, 0\}$ which expresses absolute belief corresponds to $\varphi(\infty, 0)$ or the absolute probability, and that $\pi = \{0, 1, 0\}$ which expresses absolute disbelief corresponds to $\varphi(0, \infty)$ or the zero probability. By defining φ as a function of π according to (7), the interpretation of φ corresponds exactly to the interpretation of π. The correspondence in the opposite direction is established by including Eq.(1) in (7) and solving the system for $\{b, d, i\}$ to obtain:

$$\begin{cases} b_p = \frac{r_p}{r_p + s_p + 1} \\ d_p = \frac{s_p}{r_p + s_p + 1} \\ i_p = \frac{1}{r_p + s_p + 1} \end{cases} \tag{8}$$

Eq.(8) defines a bijective mapping between Φ and Π so that any opinion has an equivalent mathematical and interpretative representation as a 2-order B-pdf and vice versa. We will call this mapping the *probability-opinion mapping.*

Definition 7. *The function* $\widehat{\Phi\Pi} : \Phi \longmapsto \Pi$ *defined by* $\widehat{\Phi\Pi}(\varphi(r_p, s_p)) = \pi_p$, *where the components of* π_p *are defined according to (8), is called the probability-opinion mapping between* Φ *and* Π.

The $\widehat{\Phi\Pi}$-mapping, which also has an inverse, will make it possible to define the equivalent of consensus and ordering of opinions.

3.6 Eccentric Opinions and 2-Order Probability Estimates

Having established the mathematical and interpretative equivalence between elements from Φ and Π, this section will describe the correspondence between elements from ${}_\epsilon\Phi$ and what we will call the class of *eccentric opinions.*

Definition 8. *Let* ${}_\epsilon\pi$ *be a linear combination of elements in* Π *such that*

$${}_\epsilon\pi = \sum_{i=1}^{n} \pi_i \delta_i, \quad n \text{ is positive integer}, \quad \delta_i \in (0, 1], \sum_{i=1}^{n} \delta_i = 1$$

then ${}_\epsilon\pi$ *is called an eccentric opinion. We will denote by* ${}_\epsilon\Pi$ *the class of eccentric opinions.*

We have not been able to conceive a graphical illustration of eccentric opinions. However they correspond to linear combinations of 2-order B-pdfs, and as such, an opinion about the proposition p : "*I will throw a "five" with the loaded dice*" is an eccentric opinion, because it can not be expressed as a simple opinion. Numerically, the eccentric opinion, which corresponds to Fig.3, can be expressed as:

$${}_\epsilon\pi_p = \{\frac{1}{2}, \frac{1}{2}, 0\} \cdot \frac{1}{6} + \{\frac{1}{10}, \frac{9}{10}, 0\} \cdot \frac{5}{6}$$

The $\widehat{\Phi\Pi}$-mapping defined above can naturally be extended to map elements from ${}_\epsilon\Phi$ to elements in ${}_\epsilon\Pi$.

Definition 9. *Let* ${}_\epsilon\varphi$ *and* ${}_\epsilon\pi$ *be elements from* ${}_\epsilon\Phi$ *and* ${}_\epsilon\Pi$ *respectively. Let* ${}_\epsilon\varphi$ *and* ${}_\epsilon\pi$ *be defined by:*

$$\begin{array}{ll} {}_\epsilon\varphi = \sum_{i=1}^{n} \varphi_i \delta_i \,, & \varphi_i \in \Phi \\ {}_\epsilon\pi = \sum_{i=1}^{n} \pi_i \delta_i \,, & \pi_i \in \Pi \end{array}$$

where n *is positive integer,* $\delta_i \in (0, 1]$, *and* $\sum_{i=1}^{n} \delta_i = 1$. *The function* $\widehat{{}_\epsilon\Phi{}_\epsilon\Pi} : {}_\epsilon\Phi \longmapsto {}_\epsilon\Pi$ *defined by* $\widehat{{}_\epsilon\Phi{}_\epsilon\Pi}({}_\epsilon\varphi) = {}_\epsilon\pi$, *where each* π_i *is defined in function of* φ_i *according to the* $\widehat{\Phi\Pi}$*-mapping, is then called the probability-opinion mapping between* ${}_\epsilon\Phi$ *and* ${}_\epsilon\Pi$.

Since $\widehat{\Phi\Pi}$ is bijective, so is $\widehat{{}_\epsilon\Phi{}_\epsilon\Pi}$.

4 Subjective Logic

The framework presented here, which consists of a subset of a general framework for artificial reasoning called *subjective logic* currently under development by the author, contains the following operations:

1. conjunction
2. disjunction
3. negation
4. conditional implication
5. consensus
6. recommendation
7. ordering

Operations 1), 2) and 3) are equivalent with the corresponding operations defined in [Bal86]. The definitions of the operations 4), 5), 6) and 7) have as far as we know not been proposed before.

4.1 Conjunction

A conjunction of two opinions about propositions consists of determining from the two opinions a new opinion reflecting the conjunctive truth of both propositions. This corresponds to the logical binary "AND" operation in standard logic.

Definition 10. *Let* $\pi_p = \{b_p, d_p, i_p\}$ *and* $\pi_q = \{b_q, d_q, i_q\}$ *be an agent's opinions about two distinct propositions* p *and* q. *Let* $\pi_{p\wedge q} = \{b_{p\wedge q}, d_{p\wedge q}, i_{p\wedge q}\}$ *be the opinion such that*

$$\begin{array}{l} \mathbf{1}.\ b_{p\wedge q} = b_p b_q \\ \mathbf{2}.\ d_{p\wedge q} = d_p + d_q - d_p d_q \\ \mathbf{3}.\ i_{p\wedge q} = b_p i_q + i_p b_q + i_p i_q \end{array}$$

Then $\pi_{p\wedge q}$ *is called the conjunction of* π_p *and* π_q, *representing the agents opinion about both* p *and* q *being true. By using the symbol "*$\wedge$*" to designate this operation, we get* $\pi_{p\wedge q} = \pi_p \wedge \pi_q$.

As would be expected, conjunction of opinions is both commutative and associative. It must be assumed that the opinion arguments in a conjunction are independent. This means for example that the conjunction of an opinion with itself will be meaningless, because the conjunction rule will see them as if they were opinions about distinct propositions.

4.2 Disjunction

A disjunction of two opinions about propositions consists of determining from the two opinions a new opinion reflecting the disjunctive truth of both propositions. This corresponds to the logical binary "OR" operation in standard logic.

Definition 11. *Let* $\pi_p = \{b_p, d_p, i_p\}$ *and* $\pi_q = \{b_q, d_q, i_q\}$ *be an agent's opinions about two distinct propositions* p *and* q. *Let* $\pi_{p\vee q} = \{b_{p\vee q}, d_{p\vee q}, i_{p\vee q}\}$ *be the opinion such that*

$$\begin{array}{l} \mathbf{1}.\ b_{p\vee q} = b_p + b_q - b_p b_q \\ \mathbf{2}.\ d_{p\vee q} = d_p d_q \\ \mathbf{3}.\ i_{p\vee q} = d_p i_q + i_p d_q + i_p i_q \end{array}$$

Then $\pi_{p\vee q}$ *is called the disjunction of* π_p *and* π_q, *representing the agents opinion about either* p, q *or both* p *and* q *being true. By using the symbol "*$\vee$*" to designate this operation, we get* $\pi_{p\vee q} = \pi_p \vee \pi_q$.

Disjunction of opinions is both commutative and associative. As for conjunction, it must be assumed that the opinion arguments in a disjunction are independent. This means for example that the disjunction of an opinion with itself will be meaningless, because the disjunction rule will see them as if they were opinions about distinct propositions.

4.3 Negation

A negation of an opinion about a proposition consists of inverting the belief and disbelief components while keeping the ignorance component unchanged. This corresponds to the logical unary "NOT" operation in standard logic.

Definition 12. *Let $\pi_p = \{b_p, d_p, i_p\}$ be an agent's opinion about the proposition p being true. Let $\pi_{\neg p} = \{b_{\neg p}, d_{\neg p}, i_{\neg p}\}$ be the opinion such that*

1. $b_{\neg p} = d_p$
2. $d_{\neg p} = b_p$
3. $i_{\neg p} = i_p$

Then $\pi_{\neg p}$ is called the negation of π_p, representing the agents opinion about p being false. By using the symbol "$\neg$" to designate this operation, we get $\pi_{\neg p} = \neg\pi_p$.

Negation is involutive so that $\neg(\neg\pi) = \pi$ for any opinion π.

4.4 Conditional Implication

The goal of the conditional implication is to determine an opinion about the conclusion of the conditional, based on the opinion about the validity of the implication and the condition of the implication. This corresponds to Modus Ponens axiom of inference in standard logic defined by:

$$p \rightarrow q \ ; \ p \ ; \ \text{then deduce} \ q,$$

meaning that if it is true that p implies q and that p is true, then q must be true.

Definition 13. *Let $\pi_{p\rightarrow q} = \{b_{p\rightarrow q}, d_{p\rightarrow q}, i_{p\rightarrow q}\}$ be an agent's opinion about the validity of $p \rightarrow q$ as a true conditional implication, and let $\pi_p = \{b_p, d_p, i_p\}$ be the same agent's opinion about the truth of proposition p. The opinion $\pi_{q/p}$ defined by*

$$\pi_{q/p} = \pi_{p\rightarrow q} \wedge \pi_p$$

is then called the conditional opinion about q by p, representing the agent's opinion about the truth of q as a function of her opinion about $p \rightarrow q$ and about p.

The justification for the conditional opinion is that both the conditional implication and the argument must be true for the conclusion to be true, and this is the same as logical conjunction of $\pi_{p\rightarrow q}$ and π_p. This definition is consistent with Prade's extension of conditional implication[Pra85] in which truth values are replaced by probabilities, so that Modus Ponens becomes:

$$\text{Prob}(q/p) \geq a \ ; \ \text{Prob}(p) \geq b \ ; \ \text{then deduce} \ \text{Prob}(q) \geq ab$$

Although our definition is more general, it reduces to Prade's definition by collapsing the ignorance and belief components into probability.

4.5 Consensus between Opinions

The consensus rule for combining independent opinions consists of combining two or more independent opinions about the same proposition into a single opinion.

Consensus between 2-order probability estimates was defined by Def.5 in Sec.3.3. The consensus rule for combining opinions is obtained by taking the same definition and using the probability-opinion mapping (8).

Definition 14. *Let $\pi_p^A = \{b_p^A, d_p^A, i_p^A\}$ and $\pi_p^B = \{b_p^B, d_p^B, i_p^B\}$ be opinions respectively held by agents A and B about the same proposition p. Let $\pi_p^{A,B} = \{b_p^{A,B}, d_p^{A,B}, i_p^{A,B}\}$ be the opinion such that*

1. $b_p^{A,B} = (b_p^A i_p^B + b_p^B i_p^A)/\mu$
2. $d_p^{A,B} = (d_p^A i_p^B + d_p^B i_p^A)/\mu$
3. $i_p^{A,B} = (i_p^A i_p^B)/\mu$

where $\mu = i_p^A + i_p^B - i_p^A i_p^B$ such that $\mu \neq 0$. Then $\pi_p^{A,B}$ is called the Bayesian consensus between π_p^A and π_p^B, representing an imaginary agent $[A, B]$'s opinion about p, as if she represented both A and B. By using the symbol $\oplus$ to designate this operation, we get $\pi_p^{A,B} = \pi_p^A \oplus \pi_p^B$.

It is easy to prove that $\oplus$ is both commutative and associative which means that the order in which opinions are combined has no importance. Opinion independence must be assumed, which obviously translates into not allowing an entity's opinion to be counted more than once

Two opinions which both contain zero ignorance can not be combined according to Def.14. This can be explained by interpreting ignorance as *room for influence*, meaning that it is only possible to influence an opinion which has not yet been committed to belief or disbelief. An opinion containing zero ignorance can only influence opinions which do contain ignorance, not the opposite, and the result will always be the total elimination of ignorance. In reality, opinions which do not include ignorance are usually counterintuitive, except in specially designed situations such as casino games with carefully controlled probabilities.

4.6 Recommendation

Assume two agents A and B where A has an opinion about B, and B has an opinion about a proposition p. A recommendation of opinions consists of combining A's opinion about B with B's opinion about p in order for A to get an opinion about p.

There is no such thing as physical recommendation, and recommendation of opinions therefore lends itself to different interpretations. The main difficulty lies with describing the effect of A disbelieving that B will give a good advice. In the interpretation we will use here, A's disbelief in the recommending agent B means that A thinks that B is ignorant about p or that B will with try to misinform with intent. As a result A is ignorant about p.

Definition 15. *Let A, B and be two agents where $\pi_B^A = \{b_B^A, d_B^A, i_B^A\}$ is A's opinion about B's recommendations, and let p be a proposition where $\pi_p^B = \{b_p^B, d_p^B, i_p^B\}$ is B's opinion about p expressed in a recommendation to A. Let $\pi_p^{AB} = \{b_p^{AB}, d_p^{AB}, i_p^{AB}\}$ be the opinion such that*

1. $b_p^{AB} = b_B^A b_p^B$,
2. $d_p^{AB} = b_B^A d_p^B$
3. $i_p^{AB} = d_B^A + i_B^A + b_B^A i_p^B$

then π_p^{AB} is called a recommendation of π_B^A by π_p^B expressing A's opinion about p as a result of the recommendation from B. By using the symbol $\otimes$ to designate this operation, we get $\pi_p^{AB} = \pi_B^A \otimes \pi_p^B$.

This operation is associative but not commutative, and opinion independence must be assumed in a chain with more than one recommending entity.

B's recommendation must be interpreted as what B actually recommends to A, and *not* necessarily as B's real opinion. It is obvious that these can be totally different if B for example defects.

It must be assumed that the trust relationships are transitive. More precisely it must be assumed that the entities in the chain do not change their behaviour (i.e. cooperate or defect) as a function of which entities they interact with. It has however been pointed out [BFL96, Jøs96] that trust is not necessarily transitive, as defection can be motivated for example by antagonism between certain entities. The recommendation rule must therefore be used with care, and can only be applied in environments where behaviour invariance can be assumed.

4.7 Ordering

Assume an agent having opinions about different propositions. The maximum of these opinions consists of selecting the opinion containing the "strongest" belief. In this way, a set of opinions can be ordered. The definition of the Max operation is obtained by using Def.6 and the probability-opinion mapping (8).

Definition 16. *Let* $\pi_p = \{b_p, d_p, i_p\}$ *and* $\pi_q = \{b_q, d_q, i_q\}$ *be an agent's opinions about two distinct propositions* p *and* q. *Let* $\pi_{p\uparrow q} = \{b_{p\uparrow q}, d_{p\uparrow q}, i_{p\uparrow q}\}$ *be the opinion which the following algorithm would return:*

```
Max(INPUT: π_p, π_q, OUTPUT: π_{p↑q})
BEGIN PROCEDURE
   IF π_p and π_q have different (b+i)/(b+d+2i) ratios
   THEN
     RETURN opinion with greatest (b+i)/(b+d+2i) ratio;
   ELSE
     RETURN opinion with the least i;
   ENDIF;
END PROCEDURE;
```

Then $\pi_{p\uparrow q}$ *is called the maximum of* π_p *and* π_q, *representing the opinion with the strongest belief. By using the symbol "*$\uparrow$*" to designate this operation, we get* $\pi_{p\uparrow q} = \pi_p \uparrow \pi_q$.

As an example, Fig.4 illustrates how π_p, π_q and π_r are ordered. The opinions π_p and π_q have the greatest $(b+i)/(b+d+2i)$ ratios, so π_r is the weakest opinion. π_p and π_q have in fact equal $(b+i)/(b+d+2i)$ ratios, but π_q has the least i so finally π_q is the strongest opinion.

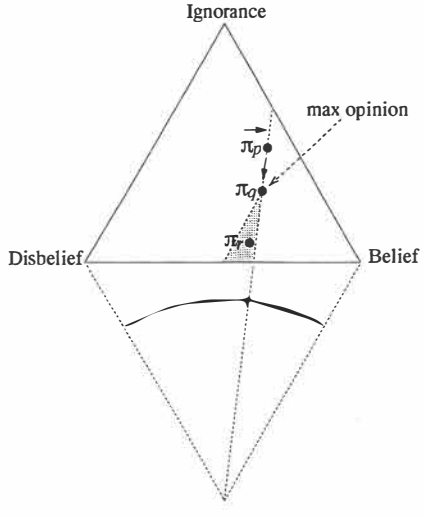


Fig. 4. Maximum of opinions

By using the symbol "$<$" for ordering opinions, we can write $\pi_r < \pi_p < \pi_q$.

It is interesting to notice that π_r's belief component in fact is greater that the belief components of π_p and π_q. The fact that π_r nevertheless is the weakest opinion can be explained by π_p's and π_q's greater ignorance which potentially can be transformed into greater belief. In other words, it is better to select

an opinion with a somewhat lesser belief component if only the ignorance component is much greater. The shaded area in Fig.4 covers the set of opinions with a greater belief component than π_q's, but which nevertheless are weaker than π_q.

Maximum of opinions is both commutative and associative. Opinion independence is not required, so that the the maximum of an opinion with itself, which is equal to itself, can make sense.

5 Subjective Algebra

5.1 Boolean Functions

We have already mentioned that conjunction and disjunction are both commutative and associative, and that negation is involutive. But not all the laws of Boolean algebra are valid with the logical operations defined above.

Conjunction and disjunction can not be combined using the distributive laws. This is due to the fact that probability estimates must be assumed to be independent, whereas distribution always introduces an element of dependence. Take for example

$$\pi_p \wedge (\pi_q \vee \pi_r) \neq (\pi_p \wedge \pi_q) \vee (\pi_p \wedge \pi_r)$$

The right side of the not-equal sign is a disjunction of two dependent opinions, because they both contain π_p. Thus only the left side represents a correct combination of conjunction and disjunction.

Conjunction and disjunction are not idempotent, because that would imply combining an opinion with itself, which would also violate the independence requirement.

However, it is easy to prove that De Morgan's laws are applicable in subjective logic. Let π_p and π_q be two independent opinions. We then have:

1. $\neg(\pi_p \wedge \pi_q) = (\neg\pi_p) \vee (\neg\pi_q)$
2. $\neg(\pi_p \vee \pi_q) = (\neg\pi_p) \wedge (\neg\pi_q)$

5.2 Mixing Consensus and Recommendation

It can be imagined that several recommendation chains produce opinions about the same proposition. Under the condition of opinion independence, these opinions generated through recommendation can be combined with the consensus rule to produce a single opinion about the target proposition.

It can also be imagined that within a recommendation chain, it is possible to obtain several recommended opinions about some of the agents in the chain. Again by assuming opinion independence, opinions obtained through the consensus rule can be recommended. An example of mixed consensus and recommendation is illustrated in Fig.5.

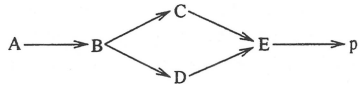


Fig. 5. Mixing consensus and recommendation

For the same reason as for conjunction and disjunction, the recommendation rule is not distributive relative to the consensus rule.

Let π_B^A, π_C^B, π_D^B, π_E^C, π_E^D and π_p^E represent the opinion relationships in Fig.5. We then have

$$\pi_B^A \otimes ((\pi_C^B \otimes \pi_E^C) \oplus (\pi_D^B \otimes \pi_E^D)) \otimes \pi_p^E \neq (\pi_B^A \otimes \pi_C^B \otimes \pi_E^C \otimes \pi_p^E) \oplus (\pi_B^A \otimes \pi_D^B \otimes \pi_E^D \otimes \pi_p^E) \qquad (9)$$

which according to the notation in Defs.14 and 15 can be written as

$$\pi_p^{A(BC,BD)E} \neq \pi_p^{ABCE,ABDE}$$

This result may seem counterintuitive at first, but the right side of (9) and (10) violates the requirement of independent opinions because they both contain π_B^A and π_p^E and thereby contain consensus combination of dependent opinions. Only the left sides of (9) and (10) thus represent correct mixing of consensus and recommendation.

5.3 Generalisation of Results

The operations described above were defined on elements from Π. It is easy to prove that they also can be applied to eccentric opinions from ${}_\epsilon\Pi$. Let for example

$$\begin{array}{l} {}_\epsilon\pi_p = \sum_{i=1}^{n} \pi_{i,p}\delta_i, \quad n \text{ is positive integer}, \quad \pi_{i,p} \in \Pi, \quad \delta_i \in (0,1], \quad \sum_{i=1}^{n} \delta_i = 1 \\ {}_\epsilon\pi_q = \sum_{j=1}^{m} \pi_{j,q}\delta_j, \quad m \text{ is positive integer}, \quad \pi_{j,q} \in \Pi, \quad \delta_j \in (0,1], \quad \sum_{j=1}^{m} \delta_j = 1 \end{array}$$

We can then for example express the conjunction of ${}_\epsilon\pi_p$ and ${}_\epsilon\pi_q$ as:

$${}_\epsilon\pi_p \wedge {}_\epsilon\pi_q = \sum_{j=1}^{m}\sum_{i=1}^{n}(\pi_{i,p} \wedge \pi_{j,q})\delta_i\delta_j.$$

All the other operations defined in Sec.4 above are also valid for elements in ${}_\epsilon\Pi$.

6 Example: Key Authentication in PGP

6.1 Brief Introduction to PGP

PGP (Pretty Good Privacy) [Zim95] is a software tool which provides confidentiality and authentication service that can be used for electronic mail and file storage applications. A crucial component of PGP is the usage of public key cryptography. We will not go into detail on how PGP works, but concentrate on the problem of key distribution and authentication. According to the PGP documentation, "*protecting public keys from tampering is the single most difficult problem in practical public key applications. It is the "Achilles heel" of public key cryptography, and a lot of software complexity is tied up in solving this one problem*" [Zim95]. We will show how our framework can be applied to enhance the efficiency and flexibility of public key distribution.

Basically, keys can be exchanged manually or electronically. For manual distribution, person A can for example meet person B physically and give him a diskette containing her public key k_A, and B can give his public key k_B to her in return. The keys can then be considered authenticated through the persons' physical recognition of each other. These keys can then be trusted and used for confidential message exchange, or for certification of other keys, as will be explained below.

For electronic key distribution, keys need to be certified by someone whom the recipient trusts for certifying keys, and who's authenticated public key the recipient possesses. For example if A possesses B's public key k_B and B possesses C's public key k_C, then B can send C's public key to A, signed by his private key k_B^{-1}. Upon reception, A will verify B's signature, and if correct, will know that the received public key of C is authentic, and can then communicate confidentially with C.

The key authentication scheme of PGP can be referred to as *anarchic*, because there exists no hierarchy of agents and no defined roles. The trust policy is based on mutual trust between pairs of agents and can be summarised as follows:

> Each agent decides individually which other agents she will trust to produce certificates. The trust level can be specified as *undefined trust*, *distrusted*, *marginally trusted*, or *completely trusted*.
>
> For a correct ownership of a public key to be trusted, it can be specified how many certificates of *marginally trusted* and *completely trusted* agents are needed. Alternatively, the agent can chose to certify the public key herself, in which case no other certificates are needed.

6.2 Implementing the Trust Policy

The trust policy and key distribution scheme of PGP can be made more general and more flexible by using the framework presented here. Trust can be specified as an opinion instead of simply *marginally trusted* or *completely trusted*. Instead of specifying how many certificates of each type is needed, the different opinions can be combined using the consensus rule, and an opinion threshold can be used to decide whether the certified key can be accepted.

Fig.6 illustrates a possible structure of public keys and their certificates possessed by A. This structure can be maintained as a relational database.

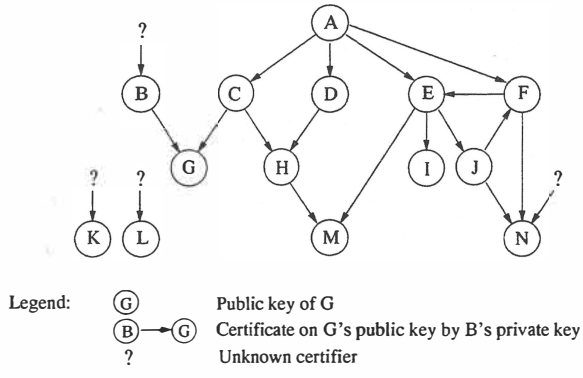


Fig. 6. Structure of public keys and their certificates.

In addition to this structure, A maintains a table of opinions about the authenticity of the public keys and opinions about their owners' trustworthiness regarding certification and recommendation, like for example Tab.1 below. Although it is not shown, a one-to-many binding between an agent and her different keys can perfectly well be accommodated within this structure.

Key k_X	Key owner X	Key Authenticity $\pi^A_{\mathrm{KA}(k_X)}$	Certification Trustworthiness $\pi^A_{\mathrm{CT}(X)}$	Recommendation Trustworthiness $\pi^A_{\mathrm{RT}(X)}$
k_A	A	{1.00, 0.00, 0.00}	{1.00, 0.00, 0.00}	{1.00, 0.00, 0.00}
k_B	B	{0.00, 0.00, 1.00}	{0.80, 0.00, 0.20}	{0.50, 0.00, 0.50}
k_C	C	{0.98, 0.00, 0.02}	{0.80, 0.00, 0.20}	{0.00, 0.70, 0.30}
k_D	D	{0.98, 0.00, 0.02}	{0.90, 0.00, 0.10}	{0.85, 0.05, 0.10}
k_E	E	{0.98, 0.00, 0.02}	{0.90, 0.00, 0.10}	{0.60, 0.20, 0.20}
k_F	F	{0.98, 0.00, 0.02}	{0.90, 0.00, 0.10}	{0.80, 0.00, 0.20}

Table 1. Table of A's opinions about public keys and their owners

It can be assumed that A knows B, C, D, E and F personally and therefore has first-hand evidence about their trustworthiness regarding certification and recommendation, so that the corresponding opinions have been determined on an intuitive basis by A. It can also be assumed that A has physically exchanged public keys with them, except for B who's key A might have received electronically but with unknown certificate. This scenario is reflected by the respective opinions about the keys' authenticity in the third column of Tab.1.

A key which is received electronically, can be considered authentic if it has been certified by someone who is considered trustworthy, and who's public key is considered authentic. There are of course other conditions, such as e.g. that the cryptographic algorithm can not be broken, but it will be assumed that these conditions are met. Let us define the propositions:

$\mathrm{KA}(k_X)$: "*My copy of the agent X 's public key is authentic*"
$\mathrm{CT}(X)$: "*I trust agent X to certify other public keys*"
$\mathrm{KR}(\{k_Y\}_{k_X^{-1}})$: "*I have received Y 's public key certified by X 's private key* k_X^{-1}"
$\mathrm{KA}(K_Y)$: "*My copy of agent Y 's public key is authentic*"

and finally the proposition

$$q \equiv (\mathrm{KA}(k_X) \wedge \mathrm{CT}(X) \wedge \mathrm{KR}(\{k_Y\}_{k_X^{-1}})) \rightarrow \mathrm{KA}(k_Y)$$

which in normal language translates into q : "*If I trust agent X to certify other public keys, and my copy of agent X 's public key is authentic, and I have received Y 's public key certified by X 's private key, then my copy of Y 's public key is authentic*".

Anyone who believes in the conditional implication q will have an opinion π_q about its validity. Whenever a certified key of an agent Y is received, and the recipient has an opinion about $\mathrm{KA}(k_X)$ and $\mathrm{CT}(X)$ regarding the certifier X, the recipient can form an opinion about $\mathrm{KA}(k_Y)$ by using conditional implication as defined in Sec.4.4.

$$\pi^A_{\mathrm{KA}(k_Y)/(\mathrm{KA}(k_X)\wedge\mathrm{CT}(X)\wedge\mathrm{KR}(\{k_Y\}_{k_X^{-1}}))} = (\pi^A_{\mathrm{KA}(k_X)} \wedge \pi^A_{\mathrm{CT}(X)} \wedge \pi^A_{\mathrm{KR}(\{k_Y\}_{k_X^{-1}})}) \wedge \pi^A_q$$

6.3 Deriving Opinions about Key Authenticity

Let agent A's opinions $\pi^A_{\mathrm{KR}(\{k_Y\}_{k_X^{-1}})} = \pi^A_q = \{1.00,\ 0.00,\ 0.00\}$, which means that they can be omitted in the calculation. Using the opinion values from Tab.1, A can for example determine an opinion about the authenticity of G's public key

$$\pi^A_{\mathrm{KA}(k_G)} = \pi^A_{\mathrm{KA}(k_C)} \wedge \pi^A_{\mathrm{CT}(C)} = \{0.78,\ 0.00,\ 0.22\}$$

which can then be used to make a new entry for G in the table. Let A's requirement threshold for accepting a key to be used for enciphering messages be $\pi^A_{\mathrm{threshold}} = \{0.91,\ 0.09,\ 0.00\}$. In this case, G's public key can not be accepted, because $\pi^A_{\mathrm{KA}(k_G)} < \pi^A_{\mathrm{threshold}}$ according to the ordering defined in Sec.4.7. B's certificate gives no additional support because B's public key has not been certified. However, H's public key will be accepted, because it is certified by both C and D. The resulting opinion about the authenticity of H's public key is:

$$\pi^A_{\mathrm{KA}(k_H)} = (\pi^A_{\mathrm{KA}(k_C)} \wedge \pi^A_{\mathrm{CT}(C)}) \oplus (\pi^A_{\mathrm{KA}(k_D)} \wedge \pi^A_{\mathrm{CT}(D)}) = \{0.92,\ 0.00,\ 0.08\}$$

6.4 Deriving Opinions from Recommendations

At least two types of recommendation can be imagined.

1. Recommending other agents for key certification
2. Recommending other agents for further recommendation

Although different opinions about an agents trustworthiness regarding the two types of recommendation can be imagined, we have assumed them to be equal so that the opinions in column 5 of Tab.1 encompass both types.

According to Tab.1, A does not trust C to recommend other agents, but she trusts D. If D recommends trusting H for certification by sending to A the signed opinion $\pi^D_{\mathrm{CT}(H)} = \{0.95,\ 0.00,\ 0.05\}$, then A can

take H's certificate into account together with E's certificate in order to form an opinion about the authenticity of M's public key. First A calculates $\pi^A_{\mathrm{CT}(H)}$ and subsequently $\pi^A_{\mathrm{KA}(k_M)}$:

$$\pi^A_{\mathrm{CT}(H)} = (\pi^A_{\mathrm{KA}(k_D)} \wedge \pi^A_{\mathrm{RT}(D)}) \otimes \pi^D_{\mathrm{CT}(H)} = \{0.79,\ 0.00,\ 0.21\}$$
$$\pi^A_{\mathrm{KA}(k_M)} = (\pi^A_{\mathrm{KA}(k_H)} \wedge \pi^A_{\mathrm{CT}(H)}) \oplus (\pi^A_{\mathrm{KA}(k_E)} \wedge \pi^A_{\mathrm{CT}(E)}) = \{0.91,\ 0.00,\ 0.09\}$$

It can easily be verified that $\pi^A_{\mathrm{KA}(k_M)} > \pi^A_{\mathrm{threshold}}$ so that it is acceptable for A. The new entries for the derived opinions are shown in Tab.2 below.

Key k_X	Key owner X	Key Authenticity $\pi^A_{\mathrm{KA}(k_X)}$	Certification Trustworthiness $\pi^A_{\mathrm{CT}(X)}$	Recommendation Trustworthiness $\pi^A_{\mathrm{RT}(X)}$
k_G	G	{0.78, 0.00, 0.22}	{0.00, 0.00, 1.00}	{0.00, 0.00, 1.00}
k_H	H	{0.92, 0.00, 0.08}	{0.79, 0.00, 0.21}	{0.00, 0.00, 1.00}
k_M	M	{0.91, 0.00, 0.09}	{0.00, 0.00, 1.00}	{0.00, 0.00, 1.00}

Table 2. New entries for derived opinions

7 Conclusion

The trust model presented in this paper is more complete than previously proposed models, and represents in fact a generalisation of standard probability calculus and logic. The model contains a varied set of operations for combining opinions, and we believe that it can be used directly for reasoning about trust in practical security applications.

References

[Bal86] J.F. Baldwin. Support logic programming. In A.I Jones et al., editors, *Fuzzy Sets: Theory and Applications*. Reidel, 1986.

[BFL96] Matt Blaze, Joan Feigenbaum, and Jack Lacy. Decentralized trust management. In *Proceedings of the 1996 IEEE Conference on Security and Privacy*, Oakland, CA, 1996.

[Jøs96] A. Jøsang. The right type of trust for distributed systems. In C. Meadows, editor, *Proc. of the 1996 New Security Paradigms Workshop*. ACM, 1996.

[Jøs97] A. Jøsang. Prospectives for modelling trust in information security. In Vijay Varadharajan, editor, *Proceedings of the 1997 Australasian Conference on Information Security and Privacy*. Springer-Verlag, 1997.

[Pra85] Henry Prade. A combinational approach to approximate and plausible reasoning with applications to expert systems. *IEEE Trans. on PAMI*, 7(3):260–283, 1985.

[RS97] Michael K. Reiter and Stuart G Stubblebin. Toward acceptable metrics of authentication. In *Proceedings of the 1997 IEEE Conference on Security and Privacy*, Oakland, CA, 1997.

[Sha76] G. Shafer. *A Mathematical Theory of Evidence*. Princeton University Press, 1976.

[Zim95] P.R. Zimmermann. *The Official PGP User's Guide*. MIT Press, 1995.

SIXTY IMPORTANT LINKS IN A COMPANY's INFORMATION SECURITY CHAIN.

Thomas Finne
Turku Centre for Computer Science/Åbo Akademi University.
Lemminkäinengatan 14A, DataCity, 20520 Åbo, Finland.
E-mail: tfinne@mail.abo.fi

Abstract: *In this paper[1] I will attempt to show how huge the subject information security actually is and present 60 important links in the Information Security Chain (ISC) that I have created.*

Key words: Information Security (ISEC), Information Security Chain (ISC), links, risks.

1. INTRODUCTION.

The *links* in the *Information Security Chain* are key factors in a company's *information security*. The information security chain is a part of a company's *total security*. The *total security* in a company has many forms and variations, for example, investment security, production security, environment security and transport security. They depend on each other and together build up the *total security*. There are many factors that affect a company's information security. We have to see the company's ISEC as an information security chain that should not have any weak links. Some parts of the ISC are more important than others but the *information security chain has to hold.* With the following example I show the subject's range.

A machine involved in production works badly, which results in a short circuit. The short circuit disturbs the distribution of electricity and the computers in the neighbouring office are damaged. Security has been damaged and economic losses can, at worst be significant. If the office had protected its computers with an UPS (Uninterruptable Power Supply), the computers would have coped with the short circuit. Thus a security issue that does not seems to have anything to do with information security actually has a great influence on the company's ISEC. The machine that malfunctions seems to be a security, or more exactly a *production security,* issue. However, the machine has also had effect on *information security*.

What are the problems in a company's information security? How should we protect ourselves against the ISEC risks? We cannot solve all problems in a company's information security but it is possible to decrease the size of the problems. However, I consider it impossible to achieve 100% ISEC; for that a human being is too erroneous and nature too insecure.[2] Each link in my ISC is treated separately as a problem, but nevertheless as a link in the information security chain. The ISC that I have created consists of *12 modules and 79 submodules.* In this paper I will deal with 60 of the modules. I have followed the research methodology, action reserach, for creating this ISC (see e.g. Arbnor et al, 1977. Checkland, 1981. Carlsson, 1991).

2. INFORMATION SECURITY DIVIDED INTO 12 MODULES.

From my figure below, it is possible to get an idea of the subject information security. The black square symbolises the company as it exists in the surrounding world. The *12* circles represent the modules that together form the information security in a company. Every module is situated partly on the outer side of the square. In this way I have tried to show that the surrounding world affects the information security of a company.

---

[1] Parts of this paper have previously been published in: Finne, T. The Information Security Chain in a Company. Computers & Security. Volume 15. Number 4. 1996, pp 297-316.

[2] Finne, T. Analysing Information Security: A Knowledge-based DSS Approach. 1996

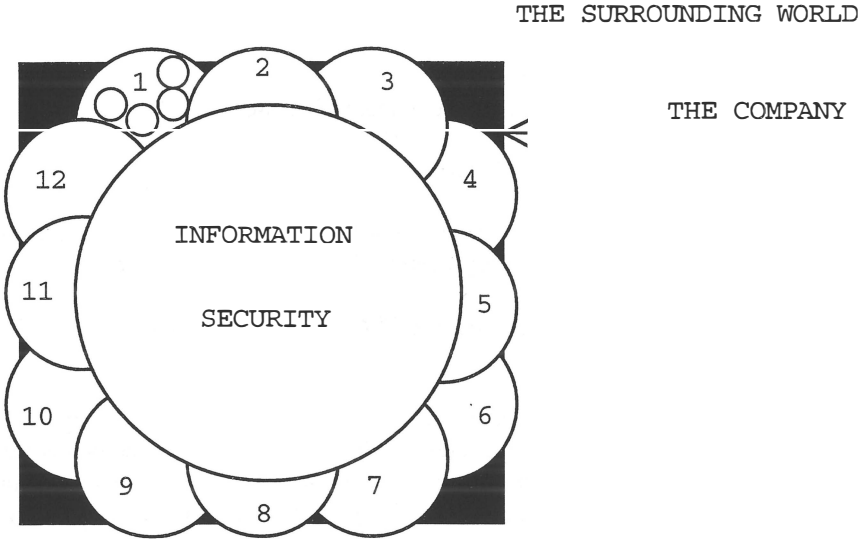


Figure.1: Information security
Source: Finne, T. In: proceedings of the IFIP TC11:WG11.2. 1996, pp 73-88.

Every module also includes many submodules with an influence on information security; this is symbolised by module number one. Thus the large circle represents the subject of information security which is made up of *12 modules and 79 submodules.* Below I will treat 60 of those submodules.

COMPUTER SECURITY

Back-up

There should be a *full back-up* system for all essential information. However, back-up is often seen as a waste of time - until one needs it.[3] There are *different media* that can be used for doing back-ups. The most common are: DAT or QIC tapes, portable disks, diskettes, VHS tapes and optical disks. I consider that QIC tape is still the most used technique. Because of malfunctioning software, hardware and human mistakes it may be that the back-up made, even over a longer period of time, has failed. Therefore it is of great importance to *test* the back-ups. It is important to find effective, reliable and user-friendly back-up software.[4] There should be a *defined cycle* for making back-ups.[5] It is important to note that all files need not necessarily be backed up, just the most important ones.

Off-site storage

In the PC Plus magazine it was pointed out that besides making back-ups, a company also has to store the back-ups in a secure place, preferably *off-site storage*.[6] The off-site back-ups should be *encrypted*.[7] There are companies *specialising* in storing companies' back-ups. In this way companies do not have to build their own storage facilities. The company storing the back-ups has to be extremely reliable.

---

[3] Maynard Jeff. Computer Audit Update. December 1994
[4] Mikrodatorn. Mikrodatorns testredaktion. Nr.1/93
Pellonpoika, P. et al. MikroPC. November 1993
Lahtinen, Tapani. PC Tietotekniikan maailma. 1995
[5] Nordic Council of Ministers. 1993
[6] Norfolk, David. PC Plus. December 1993
[7] Maynard Jeff. Computer Audit Update. December 1994

Cold and hot sites
For companies with a heavy reliance on computers so called *cold sites* can be necessary. These facilities are empty computer rooms with everything besides computers and communications systems installed.[8] For example when a fire has destroyed the computer centre it can be useful to have a cold site. Today it should be possible to get hard- and software from the distributors within a reasonable time. However, installation always takes a certain period of time. For companies with an extremely heavy reliance on computers it might be necessary to have a so-called *hot site*, which is a fully equipped "spare" computer centre.[9] Saari points out that the "spare" computer centre/room need not be as large as the original, but can be approximately 50% of the capacity of the original.[10] Both cold and hot sites can also be *shared* by many companies.

There are *movable containers* which contain ready installed hard- and software. Many companies can be partners in such a container. The container is transported to the company when needed, for example, to a company where flooding has destroyed the computer room.

Biometric methods
Biometric methods are those of fingerprint, hand geometry, face, eye, voice, signature and typing rhythms. Biometric methods, specially when *combined* with good password security, can give high ISEC. However using biometric methods introduces at least 2 new weaknesses: the database with the templates and the transmission of the biometric reading.[11] I consider that biometric methods will grow in popularity as the price of biometric instruments declines and their operational security increases.

Card access
The use of plastic cards for access to PC's is one way to improve ISEC in a company. The use of the card is usually combined with the use of a personal code. The cards can also be provided with a photo of the owner. A *log* in a microcomputer can register when a card is used. Such a card can also be used for gaining access to printers and faxes. It can also be used for external security, e.g. outer doors. There is the risk with unauthorised use of a card, e.g. if an employee loses the card. Lost cards have immediately to be blocked.

Disk-free stations
If a company has hundreds of PC's and they are connected to each other in a network, *even one user's* bad password security can lead to major ISEC breaches. This is especially the case if that user is also permitted to have external computer contacts, which increases risks. According to Lindberg a way of minimising these ISEC problems is to use disk-free stations and use passwords to get access to the server.[12] Just a few key persons would have the authorisation to *copy information* on to diskettes. I consider that the use of disk-free stations will increase.

Computer locks
At least the *server* should be provided with a computer lock in its disk station. Also the PC's can be converted to disk-free stations by installing computer locks in the disk stations. The key to the computer lock has to be kept in a safe place and must not be lost. PC's are provided with an *inbuilt lock* that can be used to shut off the PC. The user can thus improve the ISEC by locking the PC. However, there is always the risk that the key will be lost.

---

[8] Wood, Charles C. Effective Information Security Management. 1991
[9] Dorey, P. Security Management and Policy. 1991
[10] Saari, Juhani. Tietoturvallisuuden käsikirja. 1987
[11] Kim, Hyun-Jung. Computers & Security. No. 3. 1995
[12] Lindberg, Björn. Persondatorn, nätet och säkerheten. 1990

Data encryption
Encryption software packages are provided from many sources and are today used more frequently in business. According to Hoffman there are 900 cryptography hard- and software products on the market.[13] Encryption should be used much more and we might well ask why it is not used more. According to Highland[14] it is largely because of the belief that one must be a mathematician to understand encryption. I consider that encryption is a very good way to protect information (as long as the user remembers the key).

The *system administrator* normally has access to all files in a computer system. Therefore the administrator can be a great information security risk. However the risk can be minimised if the important files are encrypted. The administrator can still do his work.[15] In order to improve the security around the cryptographic key it might be necessary to *divide* the key into two parts, each part kept in the custody of a different person in the company.

Passwords
Passwords play a significant role in computer security. However, there are also many risks connected with passwords, e.g. the user might use his own name or other easily guessed passwords. Passwords should be *complicated* and contain both letters and numbers. There are also examples of how a user has used an appropriate password, but then in order not to forget it, he has stored the password on a scrap of paper under the keyboard. Users do not always follow a good password security. If a user has at least some imagination and does not use his own name, licence plates, etc., the password alone should contribute to a high level of information security.[16]

Every computer user in a company has to observe good password security. However, their password security can be checked because once a user selects a password, the system administrator can use a *password checker* to automatically check if the password is suitable. The password checker contains *a list* of common words and weak passwords.[17] There are "password checkers" used by hackers and crackers that contain hundred of thousands of words and combinations.

In a company there can be *several* servers and networks in use. Then the user has to know many passwords. This can bring problems in the *personal administration* of the passwords. More exactly it can be that the user uses *easily guessable* passwords, uses *too short* passwords, uses the *same password* for too long a period of time or *stores* them in an inappropriate way. The same password can be used for many servers but this means that there are multiple copies of the password. This *increases* the risks and the consequences of a compromise.[18] There are Access Control Packages that include passwords, logs, encryption etc. By using such a Package it can be possible to avoid using many password systems. However, as Wood[19] points out, the proper installation, usage and administration of such a package are essential.

Vendor-supplied passwords have to be changed immediately.[20] It should be stipulated how many failed sign-on or file/data access attempts that are considered to be "security violations" and therefore

---

[13] Hoffman, Lance. Encryption Policy for the Global Information Infrastructure. 1995
[14] Highland, H.J. Computers & Security. No.6. 1994
[15] LAN Vision Oy Tietoturvauutiset. 1/95
[16] Deborah Russell & G. Gangemi: In Tallberg, A. Artikelkompendium: Datasäkerhet och adb-revision. 1991
[17] Longley, Dennis. Information Security Handbook. 1991
[18] Deloitte Touche Tohmatsu. Information Protection and Client Server. 1995
[19] Wood, Charles C. Effective Information Security Management. 1991
[20] Nordic Council of Ministers. 1993

subject to investigation.[21] A problem is also that software producers normally construct their programs with user ID's and passwords but *applications built around* software packages are not always protected, e.g., decision support systems.[22]

Computer viruses

Viruses are a threat to a company's information security. A useful definition of computer viruses is the following: "*A section of code introduced into a program for malicious purposes, e.g. at some stage the inserted code will trigger a process which will, for example, eliminate files. The virus is present in a program, and when the program is run the virus writes itself into other programs in main memory or backing store. The effects of the virus can therefore extend to many users.*"[23] There are thousands of computer viruses. Moreover, there are many kinds of computer viruses: Worms, Bombs, Trojan horses and then of course, viruses.[24] A way to avoid computer viruses is always to *test* software before installing it and to avoid pirated software. *Portable PC's* can be of considerable danger. There are new viruses that even can be spread by E-mail. Borgström[25] writes that the Word virus brings companies economic losses. According to Bontchev[26] the many types and increasing number of macro viruses constitutes serious threats to a company.

Printer and Fax security

Printers should not be co-located with terminals if it is possible to take printouts of classified material.[27] Caelli et al. write that *facsimile equipment* is a an ISEC risk. Printouts are often observable by passers-by and the transmissions arrive when there is no personnel at work, except the cleaners.[28] A practical example is that of a *mutual printer*. In a company it is common that many people share the use of a printer. This means that the material printed out can be seen by many, and if the printer is not kept behind locked doors, for example, in a cupboard, there can happen considerable damage. Another considerable problem with faxes is that the sender can easily dial the wrong number by mistake. Further I consider that most managers should have their *own fax machine*. In that way avoiding that a clerk "employed" by a competing company, for example, gets access to classified material. This naturally presupposes that the managers are reliable and do not use the fax for sending out sensitive documents to a competitor.

Diskette security

Companies not using EDI (Electronic Data Interchange) may send *diskettes* to another location, for example, by post to a subsidiary. A diskette, e.g. containing important financial information, should not be sent by post. Such a procedure should be avoided since the diskette can be stolen, copied or damaged during transport. Another issue with diskettes is the problem with *storing the diskettes*. Often the diskettes are stored in diskette boxes that lie on a desk somewhere in the company, or in the best case in an ordinary cupboard. The amount of information saved on diskettes and hard disks is enormous. It happens very easily that the user *mixes* the diskettes and files by accident and in that way files can be overwritten and thus destroyed.

---

[21] Nordic Council of Ministers. 1993

[22] Finne, T. What are the Information Security Risks in Decision Support Systems and Datawarehousing ? Computers & Security. Volume 16. Number 3. 1997

[23] Caelli W. et al. Information security for managers. 1989

[24] Järvinen, Petteri. Tietokonevirukset. 1990

[25] Borgström, H. Ny Teknik/Data. 1996

[26] Bontchev, Vesselin. Possible Macro Virus Attacks and How to Prevent Them. Securenet'97 Proceedings, pp 35-79

[27] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993

[28] Caelli, William & Alan Tickle. Information Security Handbook. 1991

Distributed systems, time-sharing, outsourcing, and remote office.
Distributed systems, outsourcing, and *time-sharing* have become well-known during the last few years. These quite new things in EDP bring new information security aspects. The trend in computers has recently been moving to *distributed systems*. This means moving from traditional large computers to open client/server systems. The companies have mostly searched for savings in costs, increased flexibility and the possibility to choose.[29] However, we have to observe that distributed systems also decentralise information security. This does not necessarily mean that the ISEC is negatively affected, but that the importance of following the *ISC increases*. *Time-sharing* means that companies share computing services and in that way decrease costs. However, the information security chain becomes larger and the information security risks increase. For example, if one of the employees in one of the companies is unreliable, all the companies taking part in the time-sharing can be damaged, instead of perhaps "just" one company. It is quite common that companies *outsource* their information system; this may be economically justified but raises some questions about the company's ISEC. The resources saved by using shared premises and outsourcing can easily be lost in an information security breach. Those involved have to be *extremely reliable*.

The remote office is becoming increasingly implemented in companies. By the means of modern telecommunication employees carry out their work at home or another location. The remote office brings new ISEC risks. For example, how is classified material stored at the employee's home ? If classified material is brought home the employee should be provided with a safe for storing the documents, diskettes etc. and that means new costs. Data transmissions should be encrypted. *ISEC must not hinder* a company from carrying out a remote office operation but the ISEC questions have to be observed.

OPERATION SECURITY

Software security
Software is very much a part of a company's information security. Sherer[30] deals with failing software in her book, Software Failure Risk, what problems failing software can bring. For example, failing bookkeeping systems, failing software disturbing stock exchanges, etc. She writes that software can never be guaranteed to be 100% reliable.

Another problem that can be referred to as one of software security is when we start the new millennium (year 2000). This is a problem not only concerning large systems but also concerning smaller systems, e.g. systems used in production. It seems to be that the 'year 2000 problem' has not yet, generally speaking, been adequately recognized.

Software security is also something very much on a user level. Even popular software can have "bugs" that for a user can be very unpleasant. Software should be *tested* also by the user before being used, preferably on a *separate* computer & system.

Illegal use of software
Why does the author see illegal software as an ISEC problem? It's because illegal software *increases* the risk of bugs and viruses in the software. For a company the use of illegal software can also bring law suits and fines. Internationally there are projects in progress to identify and find those that use illegal software. It is very wrong to use pirated software.

---

[29] FISUGbytes. No 3. 1994
[30] Sherer, Susan. Software failure risk, measure and management. 1992

Spreadsheeting
Spreadsheeting is used in most companies. Spreadsheeting can be very useful for a company, however it brings also risks. Quoting Panko[31] *"..there is considerable and growing evidence that errors are relatively common in spreadsheeting"*. Spreadsheets should be thoroughly tested. Errors in spreadsheets can bring a company significant economic losses.

Data input security
When there is a lot of information to be put into, for example, a bookkeeping system, mistakes easily happen. The mistakes can then later, if not detected in time, lead to considerable economic losses. Information *input security* is very much a question of accuracy and enough time must be given to those entering the information. According to Saari[32], so called 'Data diddling' is the most usual way of committing frauds. The data (information) are changed before or while the data are put into the computer. To use payrolls to commit fraud is one of the more well-known ways. By, e.g., adding dummy personnel to the list and directing their salary to the criminal's own account the fraud can be of a great scale and in worst case go on for many years.

Data file destruction
It is important that there is someone who has knowledge about how to destroy information on files. More closely, by deleting files the user has not automatically destroyed the information but only the file's name in the "library". For example when the company *changes* its PC's to newer ones and it sells the old PC's, this can be fatal to the company's information security. The buyers of the used PC's can get access to sensitive data.

There are many ways to destroy all information in the file; one is to directly *delete* the information in the document. Another way is to *fill* the hard disks with unnecessary files and then delete them. This should reduce the possibility of recovering the files. There is also *special software* constructed for deleting files. Degaussing and scratching with sandpaper can also be ways to destroy data.

System administration
In a computer system there are different levels of users. Some are more privileged than others, i.e. the users have access to different levels of information. The *super-users* are the most privileged ones (administrators) and can change any part of the system. In Stoll's[33] famous book "The Cuckoo's Egg" the penetrator (a spy) succeeds in becoming a super-user. The system administrator is one of the key persons in a company's ISEC. That person has to be extremely reliable. Mostly the system administrator has, in one way or another, access to all computer files in a company. One possibility is to use cryptography to maintain security around the files. The network has to be properly maintained and documented. Porvari points out that the installing of network analysers and other equipment in the network should be done only by those appointed to do it.[34]

PROTECTION AGAINST BURGLARY

Guard duty
One way to prevent burglaries is to engage *guard duty*. This can be done either by engaging a firm specialising in guard duty or by employing personnel for this task. I think that guard duty is very useful for many companies but naturally a guard also brings new ISEC risks. Perhaps it is best that the guard

---

[31] Panko, R. Hitting the Wall: Errors in Developing and Debugging a "Simple" Spreadsheet Model. 1996
[32] Saari, Juhani. Tietoturvallisuuden käsikirja. 1988
[33] Stoll, Clifford. The Cuckoo's Egg. 1989
[34] Porvari, P. Pohjola-Yhtiöt. Tietokonekeskukset. 1994

will not, except in an emergency, be given access to the most sensitive parts of the company (in an ISEC meaning).

Alarms
Some offices are situated in buildings that *are not built* for use as an office and, as a consequence, many important security issues have been neglected. For example protection against fire and burglary may be neglected or undersized.

A company should have some kind of *burglar alarm*. The alarm should be *connected* to some kind of guard office or police station. Alarms signalling in the night, without anyone taking notice of them, probably have little impact on a thief. Some companies instal 'dummy' alarms to frighten possible thieves. There is also the possibility of building a *fence* around the company. The fence should be robust enough. In extreme cases such a fence can be electrified.

Television monitoring of critical areas helps to identify the intruder (often after crimes have been committed). If the cameras are wired to TV monitors in a guard office, it allows a guard to keep many critical areas under surveillance. However, observations have shown that no individual can be attentive to many monitors. Therefore television monitoring should be used in conjunction with alarms that direct the guard's attention.[35] The monitoring has to be *recorded and the tapes adequately stored.* Naturally the cameras have to be adequately maintained.

Access control
Burglaries can also have a great impact on a company's ISEC. The thieves search for cash and easily saleable goods, which also includes hardware. The direct economic losses to a company can be *considerable*, but worse is that the company can *lose information* in the form of lost back-up media and hard disks. The worst scenario is that the information stored on them is *sold to competitors*; thus the burglary can be fatal to the company. Implementing guard duty and/or alarms can diminish the risks. However, both are very expensive.

There are many ways in which access control can be carried out. In the previous section about computer security I treated access cards. According to Wood[36] far-reaching use of access control has even led to restricted access to toilets and cafeteria. Information gathered has then been used for measuring personnel performance. I think that such ISEC procedures are too heavy and might disturb the enterprising nature of the company.

Safes
A safe should be *heavy, robust and fireproof*. Such safes are very expensive, which is why companies avoid buying them. However, a good safe is important for a company's ISEC. A safe of low quality can be a big ISEC risk because it is a gathering place for very sensitive information and the employees trust that the information is adequately protected. It has happened that such a safe has been carried away by the thieves.

PROTECTION AGAINST FIRE

Alarms
Fire can be a disaster for a company's information security. For example, if a whole office building is destroyed in a fire, the losses of information in destroyed paper, diskettes, hard disks, back-up media, personnel, etc. can lead to the bankruptcy of the company. The fire itself can destroy a lot of

[35] Parker, Donn. Computer Security Management. 1981
[36] Wood, Charles. Computer Fraud & Security Bulletin. 1995

information but also different chemical mixes that emerge can be disastrous. Non-smoking regulations must be followed, especially in computer rooms and storage rooms for documents. By using alarms it is possible to rapidly detect a fire. However, as in burglar alarms, the fire alarm has to be connected to some kind of service that reacts to the alarm.

Sprinklers
Halon gas has been forbidden for use as a fire-extinguisher. Therefore water has become increasingly used in fire extinguishing. Water is effective against fire but can destroy hardware. However, hardware can sometimes be *recovered* by drying the hardware with vacuum cleaners.

Fireproof safes and cupboards
A *fireproof safe* can save a company from many problems when a fire has taken place. The most important information can be locked into a fireproof and burglar-proof safe. Less important information can be locked into fireproof cupboards. It is important to note that such safes and cupboards should also be used during the *day-time*. Fire, burglaries etc. can happen both during the day and at night.

PROTECTION AGAINST WATER DAMAGE

Building material and construction
Water leakage can be disastrous for a company. Smith writes that *internal water or sewage pipes* should not pass over any computer equipment.[37] For example, *toilets* can be a threat to a computer centre. It has happened that leaking or flooding toilets have destroyed whole computer centres. According to Raunio[38] the causes of water leakage are the quality of water, misinstallations and corrosion.

Water sensors
There are *water sensors* on the market which react to water leakage. Such sensors should at least be installed in the most sensitive parts of a company's building, e.g. the computer room and surroundings.

ELECTRICITY DISTRIBUTION

The electricity supply
Problems with the electricity distribution can be a serious threat to a company's information security. Peaks in the electricity distribution, lightning, electromagnetic interference, interruptions in electricity distribution, etc. can seriously threaten ISEC.[39] A *UPS* (Uninterruptable power supply) is a useful device that both frees the electricity from disturbances and provides the hardware with electricity, thanks to its inbuilt batteries.

Electromagnetic interference
According to Hannula[40] there can be problems when office and production facilities are in the same building. Electromagnetic interference from production machinery may damage computer equipment. *Machines* emit magnetic waves that can disturb the frequency of the electricity; especially where production and office are situated in the same building it can be a problem.

---

[37] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993
[38] Raunio, Helena. Tekniikka & talous. 19.11.1992
[39] Hannula, Antti & Risto Siilasmaa. 1991
Highland, Harold. 1985
[40] Hannula, Antti. et al. 1991

The electromagnetic interference (pulse) released in a *nuclear explosion* is enormous. If a computer centre has to function after a nuclear explosion, it has to be extremely well protected, for example, built underground and protected with a copper covering.

Magnetic fields

Magnetic fields can also damage the *computer's screen*. Electric cables create these fields and if a computer is situated near a cable the computer's screen can be damaged or at least not function well. Damaged equipment always threatens ISEC in one way or another.

Electromagnetic fields

Electronic apparatus produces electromagnetic fields that can be received at a distant place and reconstructed. This phenomenon, sometimes referred to as *TEMPEST*, has been overstated according to Smith. However, it has to be considered for very sensitive installations.[41] There is TEMPEST-proof equipment but it is expensive. A way to reduce the threat is to create an area around the VDUs and printers with no metal items. Leakage can only occur when the screen is displaying data or when data are transmitted, e.g. a printer is printing. The screens should be cleared of data as soon as possible and unnecessary printing of sensitive data should be avoided.[42] There is also equipment on the market that sends signals on such a broad frequency that all the signals the electronic equipment sends out will be covered.[43] Perhaps the best way is to include TEMPEST-hindering material in the walls during the building phase.[44] I consider that most companies do not need to take TEMPEST into consideration.

EXTERNAL AND INTERNAL THREATS

Sabotage and espionage

In an *increasingly competitive world sabotage and espionage* will probably become more common. In a company's building there are many others than just the company's employees, for example, salesmen, cleaners, consultants, computer repairmen and visitors. They can often walk around the company quite freely. Especially a salesman can have many different meetings in a company and even be left "unguarded" in an office, while the employee is busy with an urgent matter with a colleague.[45] *Both* sabotage and espionage can be performed by the company's own employee/s. Sabotage can, for example, be performed by a disappointed employee by implanting computer viruses. Espionage can, e.g., be performed by so called crackers. For example, companies performing resource-demanding research can be the target for espionage.

Public information

Espionage is not the only way to get sensitive information about a competitor. It is possible to use *legal ways to get quite sensitive information*. The information can be obtained, for example, from journalists, publications, consultants, suppliers, public services and universities.[46] A company should keep a low profile *about its assets in information technology*. A company should appoint a person especially to gather information for the company and give out information about the company.

COMMUNICATION

Telephone lines

Telephone lines can be *tapped*. Therefore in a telephone conversation the persons should follow strict ISEC. There is equipment that can be used to "scramble" the conversation so that no outsider can

---

[41] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993

[42] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993

[43] Caelli et al. Information Security for Managers. 1989

[44] Ongetta, S. Computer Fraud & Security Bulletin. August 1994

[45] Finne, Thomas. Informationssäkerhet. 1993

[46] Kelly, John. Kilpailija-analyysistä kilpailuvaltti. 1991

understand it. Specifically conversations over portable telephones bring ISEC risks. The risk is increased when using NMT portable phones. According to producers of portable telephones one should use *GSM* (Global System for Mobile communication) phones because they are much more secure to use. They transmit namely in digital form and therefore eavesdropping is very difficult.

Cable security
By connecting an illegal station to a cable it is possible to "tap" it for information. However, this depends very much on what type of cable is used. For example, *optical fibre* is very secure, but the use of double and coaxial cables should be avoided. "Wiretapping" can naturally be disastrous for a company but it can be even worse if the wiretapper *changes* the messages or puts totally *new messages* into the transmission.[47] It should be noted that "tapping" can be performed both inside and outside the company. Inside a company there is *network surveillance software* that can be used to detect illegal stations. Concerning outside cables, these can be protected *physically* on their way to the public telecommunication network.

External contact
Most companies make extensive use of the *telecommunications network*. This is, of course, necessary but it puts new demands on the company's security system; hackers and crackers may constitute a considerable security danger (see Kalakota & Whinston, 1996. Among other things they treat the security around electronic commerce. See also Cohen, F. IS Attacks: A Preliminary classification Scheme. 1997).

External telecommunication contacts can also be used when an employee carries out espionage, for example. This can be done by an *employee sending E-mail* containing classified information. The use of Internet brings many new ISEC risks.

Dial-up
The *dial-up* technique can be used for protection, which means that only some external telephone numbers have access. Hackers and especially crackers can cause major damage to a company's ISEC. It is known that most software contains so called *trap doors* that hackers have found, and used, in order to penetrate the computer systems. Hackers and crackers can also get access to passwords by *calling* the company and saying that they are one of the employees needing the password to the computer. One way to control if this is true is to ask them to send the *inquiry by fax* provided with their manager's signature. According to Clough et al.[48] there is a growing group of computer experts that use their knowledge for their own economic and national interest, more closely computer crimes and espionage.

Firewalls
Firewalls are one of the newest pieces of ISEC-improving equipment. With such a device it is possible to *control both inbound and outbound access* between internal and external networks. The authenticity of each client is confirmed before he can access a host on the private network, and all access activity is logged.[49] According to Hancock these measures are not secure enough, however.[50] It has happened that the installing of firewalls has made computer communications more slow. It is of great importance that the firewall is *installed correctly*.

---

[47] Hannula, Antti. et al. Mikrojen tietoturva. 1991
Caelli, W. et al. Information Security for Managers. 1989
[48] Clough, Bryan et al. Approaching Zero. 1992
[49] Herdan, Dov. NetSec Network Security products launched. May 1995
[50] Hancock, William. Network Security. June 1994

Mobile computing

Mobile computing has become more and more popular. *Portable computers and mobile telephones* are used more and more frequently. However, the security around them is not always the best. Portable computers can quite easily be lost, stolen or damaged. According to Computer Weekly[51], organised gangs are stealing portable computers to get access to inside information about companies. In the USA it is a growing problem.

Using mobile telephones to transfer information both by speech and computer brings new risks. According to Hardjono et al.[52] the transfer can be *eavesdropped* by, for example, an attacker maskerading as a mobile support station. *Technical problems* mean that the transmission of data can be disturbed and the information damaged en route. For example, problems in the electricity distribution of the portable computer or telephone can lead to that.

CONTINGENCY PLANNING

Emergency

A *contingency plan* should include *an emergency plan and a recovery plan*. An emergency plan reduces the damage and a recovery plan *re-establishes* the company's functions. A contingency plan improves a company's information security. Acccording to Smith the contingency plan is included in the Business Continuity Plan. It is crucial that the company has a clear understanding of which business functions are critical and which are not.[53] It is important that the contingency plan should be regularly tested. On ships the rescue plan is regularly tested; testing should also be done in a company. The results of the test should be *reported and followed up*.

Recovery

According to Johnson[54] it is important to identify the recovery time for each business function and its dependencies. The recovery plan is a vital part of a company's contingency plan. *Contracts* can be made with hard- and software suppliers so that they provide the company with the necessary hard & software in order to make as fast a recovery as possible. A fast recovery presupposes that good back-up procedures have been followed and that the *assets in hard- and software are well documented*.

PERSONNEL SECURITY

Recruiting

Personnel security is a vital part of the information security chain. As Smith points out: "*Unless personnel security is sound then all the other measures will be of little worth.*"[55] When *recruiting*, a company should investigate if the presumptive employee could be a possible ISEC risk. This is difficult to elicit but discussion with former colleagues and employers should make it possible to find out to some degree. I consider that any record of previous convictions should be demanded for key posts. *Personnel security is extremely important; however good (and expensive) the technical solutions we introduce, they can be broken by an employee's bad personal information security.*

Control of personnel

An employee normally possesses essential information that may be important for a competitor and which he may reveal either *unintentionally or for his own financial gain*. A company's most valuable asset is the employees. If this "asset" does not accept security thinking, it is meaningless to introduce

---

[51] Computer Weekly. 1995: In Computer Fraud & Security Bulletin. August 1995

[52] Hardjono, Thomas & Jennifer Seberry. Information Security Issues in Mobile Computing. 1995

[53] Smith, Martin & John Sherwood. Computers & Security. No.1. 1995

[54] Johnson, Mark. Computer Fraud & Security Bulletin. 1994

[55] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993

expensive and complex ISEC- improving instruments.[56] Kajava et al.[57] point out that a fair treatment of the employees and a good working atmosphere in the company, are basic questions in a company's ISEC.

The employees in a company have to be very careful about *what they tell about their company*. It is not unusual that even well educated people talk too much about the company they work in.[58] A considerable security breach is when *an employee leaves the firm.* Few companies trace employees who have left the firm and properly clean up access methods for the employee.[59] The Audit Commission notes that the most serious example of *computer abuse* is so called 'browsing' where a member of staff looks up an individual's personal details.[60] This information can then, for example be used as a personal competitive advantage. There have been some incidents of *inter-company rivalry* that have resulted in data and information being either destroyed or manipulated. These security issues may be even more dangerous than external attacks as they are much harder to protect against. It can be that the competition and rivalry is so fierce inside a company that some individuals use loopholes in the ISEC as a means of competition.

Those knowing most about ISEC in a company are mostly the system administrator and officer responsible for information security. Responsibilities and duties which act as *key control points* should not be carried out by one person alone. If they were illoyal to the company, the company could be hit by serious ISEC breaches.

Access to information
It is considered that in a company there should be some kind of *restrictions* on access to information. We cannot, even within a company, give access to all information. Usually this is referred to as the *'need to know'* principle, which means that an employee gets access only to that information he needs in his work. I consider that if a company wants to survive most *information* has to be available for the employees; an example of an exception can be the strategic plan of the company.

Human mistakes
It is an unfortunate fact that every human being makes mistakes. From an information security aspect this also has to be noted. How? By grading, or at least to note, the employees' level of accuracy. If the chance of getting caught is negligible and the consequences are small, the *temptation* to commit fraud can be too big.[61] If a human error is not detected by the safeguards, it can tempt employees to intentionally make the "error" for their own economic gain.

Contract employees & visitors
To perform his work a consultant has to have some information, but the contractor has to remember that hiring a consultant makes the information security chain longer and weaker. Contract employees should sign a contract of silence where they bind themselves not to reveal anything about the client. Also visitors to a company can be an ISEC risk.

Unauthorised work
It happens that employees do their *own business during time of work*. For example, they may keep the accounts of the sports club they are secretary of. Besides using valuable company time, they might

---

[56] Finne, Thomas. Informationssäkerhet. 1993
[57] Kajava, J. & J. Leiwo. Tietoturvahenkilöstö organisaatiossa. 1994
[58] Carter, Roy. Accountancy. March 1989
[59] Hancock, W. Network Security. June 1994
[60] Audit Commission. 1994
[61] Leiwo, Jussi & Jorma Kajava. Erään pienen organisaation tietorikosketjun tarkastelua. 1994

also use the company's information technology. IT is very expensive and many employees' unnecessary use can cause heavy expenditure. According to the Audit Commission much depends on which approach the company has selected. Some see the use of the organisation's IT facilities as acceptable within reasonable bounds. Others see it as akin to stealing.[62] I consider that *limited* use of company IT can be allowed, but *after working hours*. For example, using a PC to keep a *list of the members* of the local golf club should be allowed.

Staff shortage
Too small staff can be considered to be a threat to a company's information security.[63] *Mistakes* happen more easily and employees under stress can become unreliable. If there is a staff shortage, the employees will probably start regarding information security as a *time-consuming* function that *hinders* them doing their daily work efficiently and therefore hinders their career.

Theft by staff
Theft by staff can also be seen as a threat to a company's information security.[64] Usually the theft by staff stays at a low level, e.g. taking a few pencils from work. Extensive use of the company's material should absolutely be hindered. It is much up to what policy the company chooses. I consider that using the company's copying machine for taking some private copies should be allowed.

Piggy-backing
Piggy-backing refers to when an unauthorised user takes over an *active job* left alive by a previous legitimate user.[65] For example, someone is writing a document and leaves for a coffee break, without closing the work on the computer. A malevolent colleague can then make changes in the document in order to *sabotage the colleague's work.* Activating the screen saver and password can be a way to decrease this risk.

Officer appointed to be responsible for ISEC & ISEC education of the employees
There has to be someone appointed to be responsible for the company's information security. The person appointed has to be given *adequate support and resources.* The employees should be *educated in ISEC. Courses* should be arranged and *information on ISEC* should be distributed. Every company should have *a written manual on ISEC.*

Incident reporting
In a company there should be a system for *reporting* ISEC incidents or failures.[66] Suggestions for ISEC improvements could also be included in this system. If incidents are not reported, then probably no measures will be taken to prevent them, and so the same ISEC incidents can *happen many times* - incidents that have brought heavy *economic losses.*

ATTITUDES TOWARDS ISEC ISSUES

Written security policy
According to William[67] a formal information security policy is a prerequisite for a workable security programme. This requires, at a minimum, identifying the types of information requiring protection (for example, personal, trade secret, etc.) and specifying the control measures that apply to each type of information (e.g. storage, transmittal, disposal). If such a policy exists, then all information in the

[62] Audit Commission. Opportunity Makes the Thief. 1994
[63] ISO. Information Technology Security Techniques. N 962, annex A. 1994
[64] ISO. Information Technology Security Techniques. N 962, annex A. 1994
[65] Carroll, J.M. Portrait of the Computer Criminal. 1995
[66] Nordic Council of Ministers. 1993
[67] William, Perry E. Quality Assurance for information systems. 1991

organisation -not just that in a specific format (e.g. paper) will be addressed in a consistent manner. Especially concerning *new employees* a written security policy can be very necessary. According to von Solms[68] et al. the first step towards achieving good information security within a company is to ensure that the information security policy on hand is *followed, maintained and updated*. The security policy can in fact, according to Symonds, include *many policies*. One example is the network security policy. Such a written policy includes rules to maintain the security of network applications such as electronic mail and how the security of public data on the network will be managed.[69] The security policy has to be *updated* regularly.

Information security culture

A good technical solution does not automatically mean that information security is of a high level. Technology has to be used effectively and people have to consider ISEC as an important issue. It should somehow be possible to measure and record these aspects. According to Parker *"Security accountability in job descriptions and performance appraisals is the only sustainable motivator and then only if managers strongly support and carry out this provision.*[70]*"* Further Parker points out that good information security has to be awarded, for example, if employees carry out ISEC they could get some kind of gift.[71] Rewarding employees for good ISEC is a very interesting idea and appropriately carried out it should improve the ISEC.

In recent years *"company culture"* has become one of the buzz-words of the business world. A company with gaps (mostly unplanned) in its culture on information security issues will most probably have ISEC breaches, which will have significant influences on the functions of the company. Thus the employees have to take their responsibility for the company's ISEC. According to Ledell[72] education and information are the key-words for increasing consciousness about information security.

VARIOUS SECURITY QUESTIONS

Document security

Document security is a very traditional ISEC question. Due to the rapid growth in the use of computers and electronic transmitting, for example E-mail, the use of *paper should have diminished.* However, I consider that the development seems to have *gone the opposite way. The amount of paper in companies has become even greater.* Company internal reports that should be distributed to just a few persons can be distributed by mistake to many employees, etc. According to Lujanen[73] a company has to classify its information in levels. This concerns also other media than paper, more exactly, films, photos, drawings, disks etc

One way to improve document security is to follow *'the clear desk policy'*. This means that all documents have to be locked away in secure boxes etc. when not used or when the workplace is left unattended.[74] The clear desk policy diminishes the risk of someone *stealing or copying* the documents, but also the risk of documents being *lost* in, for example, a fire. If the policy is implemented, there has to be an adequate number of lockable cupboards. There should be *back-up* (copies) on documents. In case of fire the back-up should be stored off-site. Also the storage of "old" documents and files is important. Though the documents and files are not necessary in the daily life of the company, they

---

[68] von Solms, Rossouw & L.R. Meyer. Information Security Accreditation - The ISO 9000 Route. 1995
[69] Symonds, Ian. Computers & Security. No 6. 1994
[70] Parker, Donn B. Computer Fraud & Security Bulletin. July 1995
[71] Parker, Donn B. Information Systems Security. Winter 1995
[72] Ledell, Göran. Dataolyckor - har det verkligen hänt någon gång ? 1992
[73] Lujanen, Pentti. Yrityksen tietoturva. 1991
[74] Smith, Martin. Commonsense Computer Security, your practical guide to information security. 1993

have to be stored for *longer periods*. For example, *contracts* usually have to be saved for long periods. Appropriate storage media should be selected.

Temperature, dust, smoke and particles
Air pollution can also have an effect on information security. Small particles can end up between the hard disks and the reading laser head. The disk rotates very fast and can thus be destroyed. Also *temperature* can be of importance. As a common recommendation 22°C±1°C can be seen as a suitable temperature in an office building.[75] What has to be avoided are *rapid changes* in temperature. For example, starting a computer after having been in the car on a cold winter night can damage the hard disk. This occurs because the variation in temperature leads to changes in the size of material.[76] This specially concerns *portable computers* that can be left in a car in summer, for example, where temperatures may be +50°C. It is also important to note that too high a temperature and too much dust, smoke and particles in an office building have a negative impact on the working environment and thus also on ISEC. However, in my experience most computers and especially portable computers, are more robust than one expects.

Scavenging
Scavenging means searching in the *company's waste bins for information*, e.g. printouts and diskettes. Companies should have *separate* waste bins for office and other rubbish. Probably it would be best to destroy all outgoing rubbish by burning it or mechanically destroying it. Waste bins can also be used to *"send"* out information that it would otherwise be impossible to take out of the company.

Shoulder surfing
By *shoulder surfing* I mean that someone gets information by looking over the shoulder of a terminal user. Shoulder surfing can be one way of gaining easy access to a password. The user should not even allow a colleague to watch when entering the password/s.

3. CONCLUDING REMARKS

In this paper I have dealt with 60 key factors in the ISC. To notice all links, both small and large, is a demanding task. In this way we can make the chain as *strong* and *resistant* as possible. Reality is, however, unfortunately, not like the chain. For example, a company has made heavy *investment* in computer security. However, the company has just invested marginally in physical security. In this case the company has deliberately made this trade-off. Why? Because knowledge in the company has been in the computer area and the hard & software distributors have conveniently distributed and installed the ISEC products. Therefore the investment had been made in computer security. Well, it is positive that computer security is high but the poor level of physical security permits any average thief to break into the office and steal the computers and consequently also the information stored on the hard disks. Naturally ISEC is also a question about limited resources that a company works within.

Has a company to observe *all the links* (submodules) in the information security chain ? For example, companies from a specific branch of business would have to follow a smaller number of submodules. I consider that it is *up to the company* to decide what kind of ISEC they want. Only the company knows *what level of ISEC* they are satisfied with and what *investment* they can afford to make in information security.

---

[75] Mustonen, Antero. ATK-tilojen suunnittelu. 1985
[76] Hannula, Antti et al. Mikrojen tietoturva. 1991

REFERENCES

Arbnor, Ingeman & Björn Bjerke. Företagsekonomisk metodlära. Studentlitteratur. Lund. 1977

Audit Commission. Opportunity Makes the Thief. Press on Printers. London. 1994

Bontchev, Vesselin. Possible Macro Virus Attacks and How to Prevent Them. Proceedings of Securenet'97. Cannes 20-21.3.1997, pp 35-79

Borgström, H. Ny Teknik/Data. Virus blir bara värre. Vecka 17. 1996, p 4

Caelli, William., D. Longley & M. Shain. Information Security for Managers. Stockton Press. UK. 1989

Caelli, William & Alan Tickle. Communications Security: In Caelli, W., Longley, D. & M. Shain. Information Security Handbook. Stockton Press. New York. 1991, pp 649 - 706

Carlsson, Christer. Human Systems Management. New Instruments for Management Research. Volume 10. Number 3. 1991, pp 203-220

Carroll, J.M. Portrait of the Computer Criminal. In Information Security The Next Decade. Ed. J. Eloff & S. von Solms. Proceedings of the IFIP TC11 Eleventh International Conference on Information Security. South Africa 9-12.5. 1995, pp 533-545

Carter, Roy. Accountancy. Careless words cost business. March 1989, pp 158 - 159

Checkland, Peter. Systems Thinking, Systems Practice. John Wiley & Sons. Great Britain. 1981

Clough, Bryan & Paul Mungo. Approaching Zero. Faber and Faber. London. 1992

Cohen, Fred. Computers & Security. IS Attacks: A Preliminary Classification Scheme. Volume 15. Number 1, pp 29-46

Computer Weekly. 1995: In Computer Fraud & Security Bulletin. August 1995, p 4

Deborah Russell & G.T. Gangemi. 1991. In: Tallberg, Anders. Artikelkompendium. Datasäkerhet och adb-revision. Svenska Handelshögskolan. Helsingfors. 1994

Deloitte Touche Tohmatsu International. Information Protection and Client Server. Republic of South Africa. 1995, p 17

Dorey, Paul. Security Management and Policy: In Caelli, W. Longley, D., & Shain M. Information Security Handbook. Stockton Press. New York. 1991, pp 27-74

Finne, Thomas. Informationssäkerhet (Information Security). Åbo Akademi University. Finland. 1993

Finne, T. Computer Support for Information Security Analysis in a Small Business Environment. Greece. Samos. 20.5.1996. In Small System Security. Proceedings of the IFIP TC11:WG11.2. Ed. Jan Eloff. 1996, pp 73-88

Finne, T. Analysing Information Security: A Knowledge-based DSS Approach. Åbo Akademi University. Institute for Advanced Management Systems Research. Ser. A:456. 1996

Finne, T. The Information Security Chain in a Company. Computers & Security. Volume 15. Number 4. 1996, pp 297-316

Finne, T. What are the Information Security Risks in Decision Support Systems and Datawarehousing ? Computers & Security. Volume 16. Number 3. 1997, pp 197-204

FISUGbytes. Finnish Sun user group. No.3. 1994

Hannula, Antti & Risto Siilasmaa. Mikrojen tietoturva. Helsinki. Datacasa. 1991

Hancock, William. Network Security. Issues and Problems in Secure Remote Access. June 1994, pp 14-18

Hardjono, Thomas & Jennifer Seberry. Information Security Issues in Mobile Computing: In Information Security The Next Decade. Ed. J. Eloff & S. von Solms. Proceedings of the IFIP TC11 Eleventh International Conference on Information Security. South Africa 9-12.5. 1995, pp 125-134

Herdan, Dov. NetSec. Network Security products launched. South Africa. May 1995, p 2

Highland, Harold. Protecting your Microcomputer System. New York. Wiley. 1985

Highland, Harold. Computers & Security. Random Bits and Bytes. Volume 13. Number 6. UK. 1994, pp 458 - 465

Hoffman, Lance. Encryption Policy for the Global Information Infrastructure: In Information Security The Next Decade. Ed. J. Eloff & S. von Solms. Proceedings of the IFIP TC11 Eleventh International Conference on Information Security. South Africa 9-12.5. 1995, pp 50 - 63

ISO, Information Technology Security Techniques. ISO/IEC JTC 1/SC 27, N 962. 1994

Johnson, Mark. Computer Fraud & Security Bulletin. Business Recovery Planning - the Oracle Approach. UK. July 1994, pp 8-12

Järvinen, Petteri. Tietokonevirukset. WSOY. Porvoo. 1990

Kajava, Jorma & Jussipekka Leiwo. Tietoturvahenkilöstö organisaatiossa. University of Oulu. Department of Information Processing Science. Working paper. Oulu. 1994
Kalakota, Ravi & Andrew Whinston. Frontiers of Electronic Commerce. Addison-Wesley Publishing Company, Inc. USA, 1996
Kelly, John M. Kilpailija-analyysistä kilpailuvaltti. Rastor-julkaisut. Helsinki. 1991
Kim, Hyun-Jung. Computers & Security. Biometrics, is it a Viable Proposition for Identity Authentication and Access Control ? No. 3. 1995, pp 205-214
Lahtinen, Tapani. PC Tietotekniikan maailma. Mistä PC-mikro koostuu ? No.1. 1995, pp 44-46
LAN Vision Oy. Tietoturvauutiset. No.1. 1995, p 4
Ledell, Göran. Dataolyckor - har det verkligen hänt någon gång ? Studentlitteratur. Lund. 1992
Leiwo, Jussipekka & Jorma Kajava. Erään pienen organisaation tietorikosketjun tarkastelua. University of Oulu. Department of Information Processing Science. Working paper. Oulu. 1994
Lindberg, Björn. Persondatorn, nätet och säkerheten. Stockholm. 1990
Longley, Dennis. Access Control: In Caelli, William., Longley, D. & Shain, M. Information Security Handbook. Stockton Press. New York. 1991, pp 455 - 544
Lujanen, Pentti (ed). Yrityksen Tietoturva. Suomen Atk-Kustannus Oy. Myllykoski. 1991
Maynard, Jeff. Computer Audit Update. UK. December 1994, pp 15-18
Mikrodatorn. Mikrodatorns testredaktion. Säkra data med billig backup. No.1. 1993,pp 41-42, 47
Mustonen, Antero. ATK-tilojen suunnittelu. Rakennuskirja. Helsinki. 1985
Nordic Council of Ministers. Information Security in Nordic Countries. Method. Copenhagen. 1993
Norfolk, David. PC Plus. Protect and survive. December 1993, pp 300-302
Ongetta, Silvano. Computer Fraud & Security Bulletin. August 1994, pp 13-15
Panko, Raymond. Hitting the Wall: Errors in Developing and Debugging a "Simple" Spreadsheet Model. In Proceedings of the Twenty-Ninth Annual Hawaii International Conference on System Sciences. Ed. Jay F. Nunamaker, JR. & Ralph H. Sprague. JR. 1996. Vol II, pp 356-363
Parker, Donn B. Computer Security Management. Reston Publishing Company Inc. Reston. 1981
Parker, Donn B. Computer Fraud & Security Bulletin. The Mother of all Safeguards. July 1995, pp 10-12
Parker, Donn B. Information Systems Security. Security Accountability in Job Performance. Volume 3. Number 4. Winter 1995, pp 16-20
Pellonpoika, Pertti & Juha-Pekka Posti. MikroPC. Varakopioinnin vaihtoehdot. November 1993, pp 69-76
Porvari, Paavo. Pohjola-Yhtiöt. Tietokonekeskukset. S420. 1994
Raunio, Helena. Tekniikka & Talous. Vesi turmelee kymmeniä tuhansia asuntoja.19.11.92,pp 3-5
Saari, Juhani. Tietoturvallisuuden käsikirja. OTAVA. Keuruu. 1988
Sherer, Susan A. Software Failure Risk, Measure and Management. Plenum Press. New York. 1992
Smith, Martin. Commonsense Computer Security, your practical guide to information security. McGraw-Hill Book Company. London. 1993
Smith, Martin & John Sherwood. Computers & Security. Business Continuity Planning. Volume 14. Number 1. UK. 1995, pp 14-23
Stoll, Clifford. The Cuckoo's Egg. Tracking a spy through the maze of computer espionage. Doubleday. USA. 1989
Symonds, Ian M. Computers & Security. Security in Distributed and Client/Server Systems - a Management View. No. 6. 1994, pp 473-480
Von Solms, Rossouw & L.R. Meyer. Information Security Accreditations - The ISO 9000 Route: In Information Security The Next Decade. Ed. J. Eloff & S. von Solms. Proceedings of the IFIP TC11 Eleventh International Conference on Information Security. South Africa 9-12.5. 1995, pp 40-49
William, Perry E. Quality Assurance for Information Systems, methods, tools and techniques. USA. 1991
Wood, Charles C. Computer Fraud & Security Bulletin. Information Security Awareness Raising Methods. June 1995, pp 13-15
Wood, Charles C. Effective Information Security Management. Elsevier Advanced Technology. Oxford. 1991

Information Security in System Outsourcing

Tero Viiru, Kajaani Polytechnic,
Faculty of Technology,
FIN-87100 Kajaani, Finland.
E-mail: Tero.Viiru@kao.fi

Jussipekka Leiwo, Monash University,
Faculty of Computing and Information Technology,
MacMahons Road, Frackston, VIC 3199, Australia.
Email: Jussipekka.Leiwo@fcit.monash.edu.au

Abstract

Outsourcing agreement is a document to specify the relationship between outsourcing partners; service provider who offers the outsourcing service and client whose system is to be outsourced. A major success factor of the outsourcing is clear and comprehensive specification of duties and responsibilities regarding to information security. As the outsourcing agreement is juridically binding, service provider is responsible for acting accordingly. Therefore, the agreement is where security Measures should be agreed upon. As there is no clear understanding of contents of an adequate security specification in outsourcing agreement, this paper identifies the critical components and studies the issue of how to set requirements for outsourcing service providers and control enforcement of these requirements.

Keywords: Information security requirements, information systems outsourcing, information system agreements

1. Introduction

Information system outsourcing has usually been studied and justified from the financial point of view. This is a logical approach, since the major motivation behind outsourcing is usually reduction of operational cost of the system and gaining of special skills into the organization (Lacity and Hirschheim 1993). In general, outsourcing is a well understood area of information systems research (Barua 1995), but there is a special need for the consideration of information security in outsourcing. As the importance of information security as a quality factor of an information system is widely acknowledged, and outsourcing in generally emerging, maintenance of an adequate level of security becomes a major success factor in outsourcing. As security always increases the cost of operation, opportunity for outsourcing service provider to save by reducing security may become tempting.

Traditionally, only non-strategic systems have been outsourced. This is, anyhow, changing (Rao, Kichan and Chaudhury 1996; Hirschheim and Lacity 1997), and therefore the traditional assumption of guideline-based approach towards security (Kajava and Viiru 1996) is no longer

appropriate. Typically, information security methods have evolved from checklist-based methods to the risk analysis and evaluation criteria methods (Baskerville 1993, Backhouse and Dhillon 1996). Current checklist-based approaches (Kajava and Viiru 1996) are adequate when outsourcing noncritical systems, but when the importance of outsourced systems increases, more convincing provision of security of service providers is required. This leads to the specification of the major research question within this paper: Identification of factors that have an impact on information security in outsourcing and how requirements can be formulated to cover those factors. Once this is clarified, a method can be established to provide assurance that outsourced system satisfies these requirements.

The fundamental trade off in information security is between provision of security and cost of operation. In general, higher security, higher cost. Compromising security of client's system may reduce operational cost of service provider, and the cost effectiveness of the system may increase on the short run. On the long run, on the other hand, there is a great threat of losses due to violations of security exceeding the benefit gained by reduced security. Therefore, the need for security should not be underestimated and reductions in protection justified by reduced operational cost.

The fundamental information security objective for an outsourced system is maintenance the security as it was when systems were operated internally. Anyhow, as the client looses direct control of the system, there may be a need to increase the standard. As the outsourcing agreement is the main method for setting requirements and responsibilities for client, there is a clear need to identify the topics that affect the comprehensiveness of security specification in the agreement.

Outsourcing contract is a part of the information security policy of both client and service provider. Requirements for outsourcing can be specified as guidelines, risk analysis, evaluation criteria or information security model. The fundamental problem is to how to guarantee to the effectiveness of security mechanisms in outsourcing. Guidelines are the lowest and formal security models are the highest, though most limited from scope, level of assurance to the comprehensive of information security. The specificy and accuracy of security is risen from guidelines to evaluation criteria. Outsourcing security policy is based on results of risk and cost analysis before outsourcing. Outsourcing contract is a key element for information security policy in outsourcing.

From the three views towards outsourcing: juridical, administrative and technical (Douglass 1993), we shall adopt the administrative point of view. Information security requirement shall be specified very broadly to be any mechanism or procedure that client must implement to maintain or improve satisfactory level of information security on operations. Implementation mechanisms of requirements shall not be studied within this paper but the focus shall be on identification rather than detailed analysis of enforcement of requirements. The focus shall be on two first phases of outsourcing: preparation and specification of an agreement. The issues studied within this paper do not influence signing of the agreement. As the major reasons for outsourcing are usually cost reduction and gaining of skilled personnel, most of the research is focused on cost benefit analysis (Wang 1995). As these issues are well understood, there is no need to study them within this paper.

This paper starts with the identification and definition of key concepts and surveying the role of outsourcing agreement in business processes. The results are presented in section 2. Once the concepts are clarified, section 3 provides with a brief analysis of new threats that outsourced systems must face compared to traditional non-outsourced systems. Once threats have been identified, we shall provide a method to provide assurance from the enforcement of security threats in section 4. Finally, conclusions and ideas for further research shall be presented in section 5.

2. Outsourcing Agreement and Business Processes

Information systems outsourcing means partial or total transfer of the operational responsibility of an internal information system to an external supplier (Lacity and Hirschheim 1993, Richmond 1993). The main difference between outsourcing and subcontracting is the transferring of internal operational responsibilityof information system to an external supplier in outsourcing, but in subcontracting there is nothing transferred outside to the organization but external services are bought in. The second important difference is the transfer of responsibility. In outsourcing the responsibility of operations is transferred outside the organization, when in subcontracting the responsibility of operations still remains in-house and subcontractor offers only the needed services, but it has not operational responsibility for information system. Outsourcing involves two party's service provider and client. Service provider is the instance offering outsourcing service to a client willing to fully or partially outsource its information services.

Outsourcing is based on outsourcing agreement that is a document specifying information processing services that shall be outsourced, and setting requirement to the external party regarding maintenance and improvement of these services. Once requirements are established, parties must agree upon acceptable cost of these services. The more detailed the agreement, the more detailed cost analysis can be carried out. As both parties are attempting to maximize their benefit of outsourcing, the agreement is a crucial document to provide mutual satisfaction on conditions. This is, that the contractor makes profitable business and the outsourcing organization gains expected savings in operational costs (Lacity and Hirschheim 1993). To provide this satisfaction, the agreement should clearly specify rights, responsibilities and commitments of both parties. Outsourcing contract is the most important document and starting point to the outsourcing. Participants should also identify and assess information security threats related to outsourcing. Outsourcing contract belongs to the responsibility of top managers of organization, but the help of different experts should be used at this stage (Douglass 1993). At the highest level, top management is required to identify the strategic value of different systems, and to decide which systems must be outsourced, and to specify high level requirements for the agreement, that shall be further refined within the agreement process (Alpar and Saharia 1995).

The outsourcing contract can be made in stages (Richmond 1993) so that in designing stage of outsourcing contract the client represents the cost estimates of outsourcing. The client can accept or reject the draft. If client accepts the draft then client defines the needed investment. When investing planning is finished, the client calculates the final cost of outsourcing. The client either accepts or rejects referred offer. If client accepts the offer the system shall be outsourced.

An essential quality factor in the outsourcing agreement is comprehensiveness. Comprehensiveness refers to the inclusion of all relevant issues into the agreement. External specialists can be used to specify measures and metrics to control specifications of outsourced subsystems. It is also essential to include potential alterations, additions and sanctions with the developing of business actions (Lacity and Hirschheim 1993). To prepare into potential problems, it is imperative that responsibilities and rights of different parties regarding compensations and corrective actions are clearly specified and understood by both parties. Therefore, the agreement should explicitly specify all security requirements that service provider is expected to meet in implementation and operation of the system in an unambiguous manner.

Outsourcing organization should not restrict the business of client, but the businesses in which the client is involved affects to the information security of client. The expectations of the adequate level of information security may be different in client and service provider. A fundamental question becomes, how conflicts of different expectations can be solved. Organizational information security model may become a feasible solution. The guideline-based solutions may not be flexible and expressive enough to solve these problems. Recent trend in the management of information security is specification of extensive checklists, such as British Code of Practice (1996) and German IT Security Evaluation Manual (1996). These documents can act as guidelines to the security in many non-critical systems but need to be supported by more advanced methods when critical systems are concerned (Von Solms 1997).

Different systems must also be separated from each other as specified in the outsourcing agreement. Business functions can be flexible, but the level of information security must be stable or get improved when business situations change. The fundamental problem of this kind of situation is provision of assurance of the satisfactory level of information security. Thus the minimum standard for information security and requirements must be maintained.

In the agreement stage of contract issues such as how exactly contract confines, what matters should put in the contract and what is the reporting procedure of client must be considered. Outsourcing contract must define exactly its subject area, prevented threats and acceptable mechanisms against them (Richmond 1993). The question of acceptable protection measures is a critical success factor of the management of information security. As development of security enforcement mechanisms are an extremely complicated task it is essential that those responsible for information security clearly state which security enforcement mechanisms are to be used. As a general principle, publicity analyzed algorithms and implementations that have gone through public investigation and analysis by security professionals should be favored and proprietary and less understood measures avoided.

In the outsourcing agreement, information security requirements can be stated at three levels: first there is guideline-based approach, second there is risk analysis, and third there is evaluation criteria approach towards information security. As the level of information security rises from first level to third level, also cost increases. Therefore, it is essential for client to consider the expected security level of systems to be outsourced.

3. Security Threats in Outsourcing

Maintenance of adequate level of security is a fundamental problem in outsourcing since the outsourcing organization loses the direct control of information system and thus it cannot affect directly to the functioning of information system (Wong 1993). Because the responsibility of enforcement of information security is transferred to the service provider, the adequate level of information security must clear out in the outsourcing agreement.

The fundamental cause of new information security threats in outsourcing is the potential conflict of interests of outsourcing parties. As both parties are attempting to gain financial benefit, their interests may be in conflict and this may open the system to new threats. The level of information security may not be adequate, if the client cannot demand specific and required actions to be made for the maintaining of information security. As security increases operational cost, but reduce losses on the long run, reduction of operational costs by compromising security may be a tempting option. This is due to the fundamental property of information security. Benefit of security always comes from prevention of losses, not from increasing income.

The threats against system are same as in the internal operation of systems. Basically, all new threats are introduced by different interpretation of requirements set in the outsourcing agreement and the different environment and personnel. The fundamental problem is that has the client all the needed information and methods to maintain the information security. Is there any new and unknown vulnerability in the outsourced information system that must be taken into consideration? If the whole information system is outsourced, with the former personnel, rooms, softwares and hardware then the situation remains the same, but the employer is only changed. Then the threats of outsourcing can be quite easily noticed and predicted. There is also a theoretic possibility of external operators being untrustworthy, but that shall not be considered a major issue within this paper. We do agree that personnel security is an important facet of information security, but the focus of this paper shall be on threats that can be reduced by proper specification of the agreement.

The fundamental new information security threat in outsourcing is leakage of information from outsourced systems to competitors by new channels existing due to the transfer of operations to the client. For example, a failure in file system or improper destruction of printed material by service provider may lead to an unauthorized disclosure of business critical information to other businesses.

When information system is not functioning properly or there has been some other problems, such as intrusion, then client can fail report to the operations of information system and client can hope to solve problems before anybody notices them. Especially when there is a threat that client may lose the outsourcing partnership for the problems or client has to pay too high sanctions to the client and client has financial problems. It is very important to notice that the dependency of client is very high and this can be very fatal when problems occur and thus the problems must be eliminated beforehand and this is a one key point to the management of information security in outsourcing.

4. Security Process in Outsourcing

To provide first requirements and later assurance of the enforcement of these requirements, a contractor provides assurance of the security enforcement. A method for this shall be introduced within this section. We shall identify each step and analyze essential ones in detail. The following steps must be considered:

STEP 1. The organization considering outsourcing specifies their security requirements. At this phase, the organization reviews their information security analysis to update and refine requirements when the system is outsourced. Depending on the level of assurance expected, different levels of formalism are required. Typically, managerial responsibilities and pervasive security requirements can only be represented as high level statements, but technical, specific information security requirements can be represented and analyzed using tools and methods of different levels of formalism. Tools for representing technical security requirements are various. Several formal languages and notations have been developed for expressing requirements. Early access control models, such as BLP-model for confidentiality (Bell and LaPadula 1975) provided with means to specify mandatory access control requirements as tuples (s, o, r) interpreted intuitively as subject s having exactly r right to access object o. This notation was quickly proven to be inadequate, and formal, more flexible languages such as (Woo and Lam 1992) and (Jajodia et al. 1997) were proposed for coding security requirements. Access control requirements have also been successfully represented using Requirement Engineering (RE) approach (Dubois and Wu 1996) and Z notation (Boswell 1995). Additional coding mechanism include logic based on the theory of normative positions (Jones and Sergot 1992), security logics based on knowledge, permission and obligation (Glasgow et al 1992), specific notation for protection of associations (Leiwo and Zheng 1997) and enhanced system modeling notations such as security enhanced DFD diagrams (Baskerville 1988). Additionally, formal notations of, for example, (Kabasele-Tenday 1997) can code descriptions of threats. There are several alternatives for coding security requirements, and the selection should be made based on specific needs of protection of each system.

STEP 2. Contractor reviews the requirements. This is basically delivery of a report of security requirement analysis to the contractor for review. As the business of contractor is based on volume, requirements may need to be modified to meet the operational requirements of the contractor. Similarly to the specification of client's security requirements, different levels of formalism can be suggested.

STEP 3. Depending on the suggested changes by contractor, client either accepts or rejects the proposal. In case the proposal is rejected, then the client must return to step 1 and either to modify requirements or to find another contractor.

STEP 4. In case of accepted suggestions, the next step is that the contract is agreed upon, and the system implemented according to agreed security specifications. How this is

implemented is not as stated in section 1 - within the scope of this paper.

STEP 5. Contractor provides assurance of the security enforcement. The last step of the outsourcing process is where contractor provides client with the assurance that the system enforces required security features. The level of detail required for the provision of assurance must be stated in the outsourcing agreement, and depends on the level of required security. Depending on the level of formalism required, different security logics as listed before can be used. A suitable approach towards provision of assurance is according to the European Information Technology Security Evaluation Criteria (ITSEC 1992) where different levels of assurance can be required:

A) State: The contractor provides evidence that security mechanisms are implement to enforce required security features.

B) Describe: The contractor provides evidence that security mechanisms are implemented and a justification of them enforcing required security features.

C) Explain: The contractor provides evidence that security mechanisms are implemented and provides a detailed analysis of their capability to provide with expected security.

The problem is that the client and client must first agree the security methods and after that client must be capable to prove that security methods are functioning and effective, or some other trustworthy instance may do that. The proving of this capability in a trustworthy way is not simple task, and it is always hard to clarify the 'right threats'. So client can use a generally accepted, scientific proved, model for this, or some generally known and accepted security system. Evaluations are always evaluations, so the are not exact fact. A formal model that is proved effective in science and practice is a one suitable way to approach information security in outsourcing. However, it is important to notice that outsourcing can be operated in stages, and then different matters can be agreed and checked in each stage. It is important to agree a common goal for information security and agree the suitable methods for archiving this common goal.

1. Conclusion and Future Work

As a conclusion of this paper can be stated that contracting of outsourcing must be done in care. All the contracting matters are equally important, especially savings, costs and information security. The outsourcing contract can be made in stages and this is suitable for information security. There must be different mechanisms to prove the effective of security mechanisms, for example as in ITSEC. It is very important to make sure that the client can provide the needed service and maintain required level of information security. The costs and risks of outsourcing must be evaluated before outsourcing and different plans and policies are formulated. It is very important to make sure that security requirements are as clear as possible and the both sides of outsourcing can understand and agreed them. In the outsourcing agreement there must be written the minimum requirements for the information security and suitable security methods. The responsibilities and rights of the outsourcing parties should be clearly agreed upon in the

outsourcing agreement. It is imperative that accepted information security policies in outsourcing and recovery and strategic plans have been established. The actions of insourcing and changing of the outsourcing partner must be planned too. The education and inform of personnel in information security matters is important and must be regular. All the information security matters must be defined clearly in contract and understood. It is important to make sure that you are dealing with trustworthy and best possible service provider that you can have afford, the money is the keyword in outsourcing and information security. The more money you spent on security the less money you spent in business, but the security is not bad investment on the long run.

One fundamental security threat in outsourcing is the potential conflict of interest of outsourcing parties. Threats in outsourcing are basically same as in internal information system and all new threats are introduced by different interpretation of requirements in the outsourcing agreement. A one threat in outsourcing is the leak of information from outsourced system. In outsourcing the important security method is controlling and monitoring of service provider. It is important to plan, control, and agree the actions of client and service provider. For the controlling and agreeing of information security in outsourcing there can use ITSEC or CC-model. It can be stated that there is three different level of security, from lowest to highest, guidelines, and risk analysis and evaluation criteria, including information security models. If client and service provider can use methods of some formal model for managing information security, then information security is defined accuracy.

A one interesting question for future work is the management of information security in outsourcing. How different contracts and plans affect to the management of information security? Can plans for the management of information security is worked all at once or in stages? How outsourcing can be understood, a one huge process or a staged process? These are the fundamental questions for the management of information security in outsourcing.

References

Alpar, P. & Saharia, A. N., (1995): Outsourcing Information System Functions: An Organization Economics Perspective. Journal of Organizational Computing, 5(3), 197-217.

Backhouse, J. & Dhillon, G (1996): Structures of responsibility and security of information systems. European journal of information systems 5(1), 2-9.

Barua, A. & Richmond, W. B. (1995): Introduction to the Special Issue on Economics of Information Systems. Journal of Organizational Computing, 5(3), 195-196.

Baskerville, R. (1988): Designing Information Systems Security. John Wiley & Sons.

Baskerville, R. (1993): Information Systems Security Design Methods: Implications for Systems

Development. ACM Computing Surveys, Vol. 25, Num. 4, 375-414.

Bell, D. E. & LaPadula, L. J. (1975): Secure Computer Systems: Mathematical Foundations and Model. MITRE Corporation Technical reports M74-244. Bedford, MA, USA.

Boswell, A. (1995). Specification and Validation of a Security Policy Model. IEEE Transactions on Software Engineering, Vol. 21, Num. 2, 63-68.

Code of Practice for Information Security Management (1995), British Standards Institute Standard BS 7799, UK, 1995.

Douglass, D. P. (ed.) (1993): New Wrinkles in Outsourcing, I/S Analyzer, September 1993, Vol. 31, Num. 9, 1-17.

Dubois, E. & Wu, S. A. (1996): Framework for dealing with and Specifying Security Requirements in Information Systems. Proceedings of the IFIP TC11 12th International Conference on Information Systems Security (IFIP/Sec'96). Samoa, Greece.

Glasgow, J., MacEwen, G. Panangaden, P. (1992): A Logic for Reasoning about Security. ACM Transactions on Computer Systems, Vol. 10, Num. 3, pp. 226-264.

Hirschheim, R. & Lacity, M. C. (1997): Information System Outsourcing and Insourcing: Lessons and Experiences. Proceedings of the 1997 Pacific Asia Conference on Information Systems. Brisbane, QLD, Australia.

IT Baseline Protection Manual (1996), BSI, Germany, 1996.

Information Technology Security Evaluation Criteria (ITSEC). (1992): Provisional Harmonized Criteria, version 1.2. Commission of the European Communities COM (92), 298 final. Brussels, Belgium, September 1992.

Jajodia, S., Samarati, P. & Subrahmanian, V.S. (1997). A Logical Language for Expressing Authorizations. IEEE Symposium on Security and Privacy.

Jones, A.J.I. & Sergot, M. (1992): Formal Specification of Security Requirements using the THeory of Normative Positions. Computer Security - ESORICS'92. Springer-Verlag LNCS 648.

Kabasele-Tenday, J. -M. (1997): Specifying Security in Composite Systems. Proceedings of the 1997 Information Security Workshop. Ishikawa, Japan.

Kajava, J. & Viiru, T. (1996): Delineation of Responsibilities regarding Information Security during an Outsourcing Process from then Client's Point of View. Twelfth International Conference on Information Security, Sec '96/WG 11.1, Information Security Management in a Distributed Environment, Pythagorean, Samos, Greece, 20. May 1996.

Lacity, M. C. & Hirschheim, R. (1993): Information Systems Outsourcing. John Wiley & Sons, Guilford, Surrey.

Leiwo, J. Zheng, Y. (1997). A Framework for the Management of Information Security. Proceedings of the 1997 Information Security Workshop. Ishikawa, Japan.

Oltman, J. R. (1990): 21st Century Outsourcing. Computerworld, April 16.

Rao, R., Kichan, N. & Chaudhury, A. (Guest editors)(1996): Information Systems Outsourcing. Special Issue in Communications of the ACM, Vol. 39, Num. 7, 27-54.

Richmond, W. B. & Seidman, A. (1993): Software Development Outsourcing Contract: Structure and Business Value. Journal of Management Information Systems, summer 1993, Vol. 10, No. 1, 57 72.

Schneier, B. (1996), Applied Cryptography, 2nd edition. John Wiley & Sons, New York.

Von Solms, R. Can Security Baseline Replace Risk Analysis In Proceedings of the IFIP TC11 13th International Conference on Information Systems Security. Copenhagen, Denmark, 1997.

Wang, E. T. G. & Barron, T. (1995): The Decision to Outsource IS Processing Under Internal Information Asymmetry and Conflicting Objectives. Journal of Organizational Computing, 5(3), 219-253.

Wong, K. (1993): Outsourcing IT-Safeguarding Your Legal Interests, Purchasing & Supply Management, December, 30-33.

Woo, T.Y.C. & Lam, S.S. (1992): Authorization in Distributed Systems: A Formal Approach. Proceedings of 1992 IEEE Symposium on Research in Security and Privacy.

IT Security Awareness - Issues for Industry

Jorma Kajava and Mikko T. Siponen

University of Oulu, Department of Information Processing Science, Linnanmaa, FIN-90570 Oulu, FINLAND,
E-mail:{Jorma.Kajava, Mikko.T.Siponen}@oulu.fi

ABSTRACT

IT security awareness is a factor whose significance for the overall security level in any organization should not be underestimated. Awareness is a key issue, because it is, in essence, a preventive measure aimed at establishing correct security procedures and security principles firmly in the minds of all employees. This is important because each individual security technique can be misused or misinterpreted, thereby losing its usefulness. The tools and methods of any awareness package should be based on increasing the level of awareness of all employees. These tools include training, education and campaigning along with consistent and effective IT security management. Finally, measuring is needed to ascertain that the results of the awareness program are the intended ones and that the development effort is on the right track, heading in the right direction.

Keywords: Information Security Awareness, IT Security Education, IT Security Management

1. INTRODUCTION

A number of papers presented at recent IT security seminars have championed IT security awareness as the best preventive measure against the biggest weakness in all systems: the human factor [Ceraolo, 1996]. Industrial life, however, seems to hold a different view on the value of awareness: IT security awareness is only too often interpreted either as a trivial issue or – at best - a necessary evil. Perhaps the reason behind this sort of attitude can be traced back to the non-technical nature of the function: awareness falls outside the scope of the engineering sciences. But things are changing, and awareness is no longer an irrelevant issue. In fact, it could be stated that the importance of the awareness factor should form the basis of the security strategy of any organization [Ceraolo, 1996; Thompson & von Solms, 1997; von Solms, 1997; Kajava & Siponen, 1997 (a, b)].

Information security denotes a state of affairs that has been reached when data and information, on the one hand, and systems and services, on the other, are adequately protected both in normal times and in times of crisis. Components of an information system must be protected against various kinds of threats, harm and damage caused by hardware or software failures, acts of God, and accidental or deliberate human acts or omissions [Royal Canadian Mounted Policy, 1981]. In other words, IT security has to satisfy the criteria of the so-called CIA-dimension proposed by [Parker, 1981]. This cannot be done on the basis of purely technical solutions; a more holistic view regarding the methods and functions of IT security is needed in response to increasing threats. Available methods of protecting different system components can be divided into three categories [Kajava & Siponen, 1997 (b)]:

1. management methods, including leadership methods,
2. technical methods, and
3. support and awareness methods.

All these three functions stand for aspects of security that are important to take into consideration in formulating an organizational IT security strategy. In terms of awareness, the possible lack of appreciation by the security people of an organization is a vulnerability. Social engineering, a commonly used hacking method these days [Miettinen, 1995], is a case in point. Social engineering is a special intruder strategy which exploits an existing lack of security awareness to gain unauthorised access into a computer system [Kajava & Siponen, 1997 (b)]. Thus it could be said that IT security awareness plays a significant role in the struggle against social engineering, because it correlates directly with the amount of social engineering cases.

In this paper, in addition to re-emphasizing the importance of awareness, we also want to discuss awareness in relation to certain relevant issues in industrial organizations. In this undertaking, we shall concentrate on the following questions:

- how IT security awareness is currently understood, and how it is implemented in practice;
- what are the premises and principles of IT security awareness;
- how to measure the efficiency and functionality/dysfunctionality of IT security awareness;
- how IT security awareness could be implemented in industrial organizations in a more efficient and relevant way.

2. THE AWARENESS FACTOR

After the significance of awareness was recognised in the early 1990's, the question of how to meet this challenge was raised. We have found that there are different kinds of dynamic problems closely related to misunderstanding the significance and real nature of awareness within the overall security management of organizations. For example, people working in a particular organization may have recognized the importance of awareness as such, while failing to adequately understand its managerial dimensions or its real nature and premises. For instance, the IT security people of an organization may have totally internalized the idea of awareness and even got the approval of the top management for what they are doing. However, if they fail to recognize the role of education or training strategy, their target mission may suffer and the target goal may not be internalized by the target persons, i.e. the end-users. This is likely to be the case if the educational strategy of an organization, for example, is based on some such policy as "do not ask – just follow these printed security guidelines strictly", or some consultant comes in and orders or coerces the employees to follow a set of given security guidelines. Even when the task of raising the users' awareness is undertaken in good faith and the people involved are fully aware of the importance of following the security guidelines, this kind of approach is unlikely to be successful. There are much better alternatives to raising the awareness factor than those mentioned above.

2.1 Development of the awareness factor

There is much more to raising the level of security awareness in an organization than just arranging training sessions or passing around circulars, designed in the hope that the members of the organisation would then strictly follow the given instructions [Kajava & Siponen, 1997 (a)]. Particular attention has to be paid to the actual form (the framework) and contents of the awareness programme when building a holistic formal approach. The programme is not likely to succeed if these two factors are not at an adequate level.

Three areas having a positive effect on security education within the learning domain have been set up by [Snoyer & Fisher, 1993]:

- knowledge is improving;
- skills are developing;
- attitudes are changing for the better.

2.2 New fundamental issues of awareness

At the Sec'92 (IFIP TC-11 security conference), McLean stated that learning is the key to raising information security awareness, as it enables us to effect behavioural changes. McLean maintains that 'IT security awareness', as it is generally presented in literature, does not necessarily include learning processes [McLean, 1992], even though it should. We agree with this view, but argue further that learning merely constitutes a minimum requirement. However, although scholars like McLean regard awareness as the primary factor within IT security, the status of the awareness factor as field of research has not increased outside the USA.

On the basis of a series of practical studies, we have come to the conclusion that there are several stages in how people respond to awareness. There seem to be three stages of awareness that should be taken into account. We maintain that these stages constitute an implication relation, where - within practically every organization - there are people at every stage, and the success or failure of IT security awareness correlates with either progress upwards or a relapse downwards. The stages of IT security awareness are [Kajava & Siponen, 1997 (a)]:

1. drawing people's attention to security issues,
2. getting user acceptance,
3. getting users to learn and internalize the necessary information security activities.

The first stage includes drawing people's attention to information security-related issues and trying to catch their interest. The second stage involves user acceptance; having got the end-users' attention, it is important to get them to accept the IT security policy of their organization. Finally, at the third stage, the end-users should have internalized the instructions they have received during their security-related education and they should take corrective measures in accordance with the organization's security policy. In this paper, the term 'awareness' includes all the aspects mentioned above. [Kajava & Siponen, 1997 (a)].

IT security awareness should be comprehensive, well-organized and systematically executed from the start. In addition, the efficiency of all actions should be measured to ensure the on-going development of the organizational IT security awareness programme. As for our own IT security awareness programme in university environment, we have come to the realization that a wide range of tools and methods are needed to implement security awareness for different people in different environments and at different stages of awareness. [Kajava & Siponen, 1997 (a)]. Security education is needed to convince every user of the importance of following the guidelines, and to make users aware of the consequences of intentional violations of information security [Straub et al., 1992]. Education is also needed to ensure that the achieved level of awareness (as defined above) is maintained. Various awareness raising methods, such as campaigning and the so-called Hammer theory, are needed to provide incentives for end-users and to refresh the importance of these factors in the minds of people [Kajava & Siponen, 1997 (a)]. And finally, awareness, comprising education and training, should ensure that people internalise security guidelines and abide by them in their daily work.

3 AWARENESS PROGRAMME

The awareness programme of any industrial organization should follow the following framework developed by [NIST, 1995]:

- Identify Programme Scope, Goals and Objectives.
- Identify Training Staff.
- Identify Target Audiences.
- Motivate Management and Employees.

- Administer the Programme
- Maintain the Programme.
- Evaluate the Programme.

The awareness programme should be identified for at least four different target groups. These are: top management, IS management, end-users [Thompson & von Solms, 1997] and IT/IS specialists. Of course, there is no exact formal classification for each group and, as a result, the end-user group, for example, remains relatively vague.

The IT security awareness programme should be implemented at all levels of the organization, starting from the top management who should be made aware of the need to establish and maintain an organizational security policy [ISO-IEC-27, 1995]. Then comes the creation of a security model including IT security policies, the allocation of responsibilities, etc. [Kajava & Siponen, 1996]. IT security awareness programmes are essential in keeping users in the "security team" and in ensuring the overall success of the organization's security strategy [Curran, 1996]. All evidence shows that, to function satisfactorily, a security programme must find support in all parts of the organization [Wood, 1982]. Moreover, the top management has to accept security issues whole-heartedly and allocate resources and appropriate financial support to IT security. As stated earlier, the effectiveness of the measures taken during an awareness programme should be measured as objectively as possible. The problem is that most organizations do not provide feedback or measure the success of their IT security awareness programme [McLean, 1992 (European Security Forum, 1991)]. The security management should react to feedback and consider necessary improvements. Feedback should be based on organizational and end-user viewpoints and on the established results of particular security measures [Kajava & Siponen, 1997 (a)].

The educational dimensions of an security awareness programme should include at least the following points [ISO-IEC-27, 1995]:

- objectives behind the corporate IT security policy as well as an explanation of the policy, guidelines and directives,
- a risk management strategy leading to an understanding of risks and safeguards,
- IT security programme/plan to implement and check safeguards,
- basic information protection needs,
- establishment of a classification system concerning the protection of information,
- policy of reporting and investigating breaches of security or attempts thereof,
- implications of security incidents for end-users and the organization,
- procedures, responsibilities, job descriptions,
- security audit/compliance checks,
- change and configuration management, and
- consequences of not acting in authorized manner (including disciplinary actions).

In large companies, the responsibility of increasing corporate IS awareness should be entrusted to the corporate IT security officer [ISO-IEC-27, 1995], and the awareness programme should be accepted by the management IT security forum [Code of Practice, 1993]. Various communication channels spreading information about security education should be considered. The following information sharing channels have been proposed by [Spurling, 1995]:

- instruction booklets;
- newsletters;
- multi-media packages;
- screen savers.

These channels could be used for different purposes from imparting information to refreshing the importance of IT security issues in the minds of people.

4. METHODS OF AWARENESS

McLean has talked about selling information security to people through campaigning. This kind of action could prove very useful in terms of security education, and provide a positive impetus to information security, as it would make people more likely to remember the importance of security. On the other hand, security campaigns, like their political and advertising counterparts, could also give rise to negative feelings or even hate [Kajava & Siponen, 1997 (a)]. Hence, an awareness programme cannot be based purely on campaigning, rather, a more holistic view with particular measures is required.

Another method similar to campaigning is the so-called Hammer theory, which is based on making information security an "in" topic within an organization. The Hammer theory states that when a new concept is properly introduced in the organization, everybody is keen to use it. [Perry, 1985]. The Hammer theory and campaigning work together quite well [Kajava & Siponen, 1997 (a)].

5. IT SECURITY AWARENESS FRAMEWORK FOR INDUSTRY

All organizations should have one or more on-going awareness programmes. These IT security awareness programmes should be implemented at all levels of the organization, starting from the top management who should be made aware of the need to establish and maintain an organizational security policy [ISO-IEC-27, 1995]. Then comes the creation of a security model including IT security policies, the allocation of responsibilities, etc. [Kajava & Siponen, 1996]. The awareness programme should meet the social requirements stipulated by the corporate culture of the organization in question, because only by understanding and respecting human factors we can gain the acceptance of the employees. This is a vital factor in the equation and eventually puts us in a position to effectively improve the security of information within the organization. [Kajava & Siponen, 1997 (a)].

The goals and scope of the programme have to be identified clearly. There are at least four different target groups, requiring different kinds of security education. The top management needs to understand why IT security (and awareness) is such an important factor. If they do not share this view about the importance of IT security and awareness, it is hard to get their whole-hearted commitment, which is an essential prerequisite of success. Furthermore, it is important that the top management understands information security in general and the various threats that could have an impact on their organization [Thompson & von Solms, 1997], because all IT security actions take place under the guidance or control of the top management. Furthermore, the top management and IT security management (or board) also have to ensure that the essential parts of the organization's assets and information systems are in fact protected. This can be achieved, for example, by prioritizing them.

The IT management may need to be informed about the following issues [GMIT, 1996 (a)]:

- assignment of security roles and responsibilities;
- selection of risk analysis strategy option[1];
- IT security recommendations, including the following:
    - determination of acceptable risks,
    - selection of safeguards to reduce risks to an acceptable level,
    - benefits of the cost of safeguards vs. the reduction of risks,
    - acceptance of residual risks,
- issues concerning organizational IT security policy;
- issues concerning IT security plans;
- implementation of safeguards;
- follow-up to ensure that the safeguards are functioning properly;
- ensuring that the acquired HW & SW satisfies all the information security needs of the organization;
- contingency plan.

New employees should participate in security education and "correct" procedures should be established from the beginning. Moreover, if there is a chance of motivating ADP-experts in other fields to chip in and participate in the awareness programme, the possibility should be exploited [Kajava & Siponen, 1997 (a)].

Staff that could function as educators or trainers needs to be identified and their capability and motivation ensured. In addition, their responsibilities have to be laid down clearly. In security education, awareness stage three, the real goal of all activities, must be borne in mind at all times. In this respect, the topics outlined in chapter 3, i.e. the educational dimensions of awareness programmes, are highly relevant. Furthermore, any awareness programme should include information about intrusion through social engineering [Kajava & Siponen, 1997 (b)]. But the main message that all awareness programmes should drive home is that security is based on

[1] E.g. Baseline, Informal, Detailed, Combined.

everyone observing the correct procedures in any given situation. This, along with functional technical protection mechanisms, is the most important precaution against all forms of intrusion [Kajava & Siponen, 1997 (b)]. To cut a long story short, the importance of maximizing motivational factors cannot be underestimated.

Moreover, the programme has to be maintained, evaluated and developed on an on-going basis. The results of any form of security awareness measure should be measured in an objective way. The results of the measures provide essential feedback for assessing the success of the awareness programme.

6. MEASURING IT SECURITY AWARENESS

Generally, measurements of IT security awareness can be approached from two different angles. First of all, organizations have to measure, verify and validate the formal part of their IT security awareness programmes. These awareness programmes must be based on some kind of holistic framework and all functions and actions must be appropriate to the situation. In addition to the issues outlined above, appropriate standards such as the Code of Practice, should be used as a security checklist.

The second dimension comprises the contents of the programme, i.e. how will we know that our formally efficient awareness programme will really be as effective as it should be. Current standards do not provide much help in verifying the quality of the content of awareness programmes.

Having satisfied the mentioned dimension of measuring, we go on the other perspective, where we state that by measuring information security awareness we try to study and verify the effectiveness of our work and the results of our security measures. The findings will then form the basis for decisions about changes or improvements regarding the awareness programme. But measuring information security is not a straight-forward matter: on one hand, there are diverse methods producing absolute numerical results and, on the other hand, it could be argued that such measurements are an impossibility as information security is largely concerned with subjective feelings. In this respect, measuring information security is not unlike measuring the quality of information systems. [Kajava & Siponen, 1997 (a)].

Furthermore, we argue that the gut feeling of expert security auditors is as important as any numerical results, because there is much more to information security than mere numbers - IT security can be described as the mental state of a specialist and it includes technical, legal and social rules, heuristics and ethical approaches. Information security staff has to gather quality measurements on the effectiveness of information security management so that they can rest assured that they are making the most of their investments. [Kajava & Siponen, 1997 (a)].

There are four questions that must be taken into account when evaluating the effectiveness of information security awareness (modified from [Walsh, 1996]):

1. Did the employees like it?
2. Did the employees learn it?

3. Did the employees apply it correctly back on the job?
4. Did the training have an impact on the bottom-line of the organization?

To satisfy the basic requirement of the first question, target persons have to feel positively about the programme. If they do not, it is quite obvious that they will not be particularly keen to follow and internalize the various aspects of the programme. As a result, the whole foundation of the programme is far from solid. As to the second question, if the target persons do not learn what they are supposed to learn, they have nothing new to apply in their daily work. The problem with the third question is that even if the target persons do learn the right attitudes, procedures, etc., there are no guarantees that they will apply them correctly and strictly back on the job. Indeed, as our hypothesis about the three stages of awareness indicates, the target persons must reach the last stage to provide a positive answer to the third question. According to McLean, due to the difficulties in measurements, incident reporting could be employed as a measure of success. One way of doing this (the first method) is to count the number of misuse cases before the initiation of an awareness programme and again after a certain period of time has passed since the start of the programme. If IT security practices improve as a result of the awareness programme, then it would be expected that the number of reported incidents would decrease. However, another effect of increasing awareness will be to extend the population's judgement of what constitutes reportable events and therefore people may tend to report events which would previously have escaped attention [McLean, 1992].

The second measuring method is based on detecting instances of misuse committed by participants of the pilot awareness programme and comparing the figure with that of groups that have not received any information security education. The third measuring method involves making a comparison between different companies, namely, those that provide security education to their employees and those that do not. An IT security awareness programme and functional hardware and software security solutions tend to have a deterring effect on misuse, as the end-users know that misuse will eventually come to light. As a consequence, the level of IT security misuse tends to decrease. [Kajava & Siponen, 1997 (a)]. The problem with both the second and third approach is that there may not be an objective way of making the measurements. Moreover, in the last approach, it could prove difficult to find two organizations that are similar enough to make such a measurement valid. Informally, to overcome these measuring problems, a big organization could use the different internal groups or branch offices as a basis for comparison.

7. CONCLUSIONS

Any attempt to reach an adequate security level must necessarily build on IT security awareness. The creation of awareness requires a methodical approach, such as that provided by a systematical awareness programme using different tools and methods for increasing security awareness. These tools and methods include training, education, campaigning and IT security management. Obviously, to be effective, the methods must be used consistently. To increase end-user awareness of security related matters, the content of the programme must also come under serious consideration. The awareness programme should meet the social requirements stipulated by the

corporate culture of the organization in question, because only by understanding and respecting human factors we can gain the acceptance of the employees. Moreover, the results of security education should be measured as objectively as possible to verify whether the awareness programme has met its goals.

REFERENCES

A Code of Practice for Information Security Management, (1993), Department of Trade and Industry. DISC PD003. British Standard Institution, London, UK.

Ceraolo, J.P., (1996): Penetration Testing Through Social Engineering. Information Systems Security. Vol 4, No 4. Winter 1996.

Curran, Terri, (1996): Implementing Succesful Security Awareness Programs. 23 rd Annual Computer Security Conference and Exhibition, CSI, November 11 -13, Chicago. Il.

ISO/IEC JTC1/SC27, (1995): Guidelines for the Management of IT Security (GMITS).

Proposal for a Council Recommendation on Common Information Technology Security Evaluation Criteria (ITSEC) (1992). COM (92) 298 final. Office of Official Publications of the European Communities. Brussels, Belgium, 10 September.

Kajava, J., Leiwo, J., (1995): Information Security for Workstations - Implications for End-Users. Eleventh International Conference on Information Security IFIP-TC 11 (Sec'95/WG 11.1), Information Security Management - The Next Decade. 8 - 12th May 1995, Cape Town, South Africa.

Kajava, J., Siponen, M.T. (1996): Security Management in Organizations - Bottom-Up or Top-Down Approach?" NORDSEC '96 - Nordic Workshop on Secure Computer Systems (ed. Erland Jonsson), SIG Security and Chalmers University of Technology, Department of Computer Engineering, 7 - 8th November, Gothenburg, Sweden.

Kajava, J., Siponen, M.T. (a), (1997): Effectively Implemented Information Security Awareness - An Example from University Environment. Proceedings of IFIP-TC 11 (Sec'97/WG 11.1). 13th International Conference on Information Security: Information Security Management - The Future. 13th May 1997, Copenhagen, Denmark.

Kajava, J., Siponen, M.T. (b), (1997): Social Engineering - IT security threat of Informatics. In Kristin Braa & Eric Monteiro (eds.): "Social informatics". Proceedings of IRIS 20th June. Department of Informatics, University of Oslo, Norway.

McLean, Kevin, (1992): Information Security Awareness - Selling the Cause, In Gable, G., Caelli, W., Ng, F., Ranai, K., Soh, C, (eds.): Security and Control: From Small Systems to Large. Proceedings of the IFIP TC11 /Sec'92, Singapore, 27-29 May.

Miettinen, Juha, E., (1995): Survey of Hacking in Finland in the 1990's - Summary of the results, Research papers A 24, December 1995, University of Oulu, Finland.

The NIST handbook, (1995): An Introduction to Computer Security, NIST special publications in October 1995.

Parker, Donn, B., (1981): Computer Security Management, Prentice Hall, Reston, USA.

Perry, William E., (1985): Management Strategies for Computer Security. Butterworth Publisher, Boston.

Royal Canadian Mounted Policy (1981): Security in the EDP Environment. Security Information Publication, Second Edition. Gendarmere Royale du Canada, October.

Snoyer, Robert S. & Fischer, Glenn A., (1993): Managing microcomputer security. Edited by Robert S. Snoyer and Glenn A. Fischer. Chantico Publishing Company Publication Homewood, Ill., Business One Irwin.

Solms von, R., (1997): WG11.1 chairman's Foreword. Proceedings of IFIP-TC 11 (Sec'97/WG 11.1). 13th International Conference on Information Security: Information Security Management - The Future. 13th May 1997, Copenhagen, Denmark.

Spurling, Phil, (1995): Promoting Security awareness and commitment. Information Management and Computer Security, Vol. 3 No. 2 1995.

Straub, Detmar, Carson, Patrisia, Jones, Elizabeth, (1992): Deterring Highly Motivated Computer Abuses: A Field Experiment in Computer Security, In Gable, G., Caelli, W., Ng, F., Ranai, K., Soh, C, (eds.): Security and Control: From Small Systems to Large.

Thompson, T.E & Von Solms, R., (1997): An effective information security awareness program for industry. Proceedings of IFIP-TC 11 (Sec'97/WG 11.1). 13th International Conference on Information Security: Information Security Management - The Future. 13th May 1997, Copenhagen, Denmark.

Walsh, Tom, (1996): Measuring the Effectiveness of Computer Security Training. 23rd Annual Computer Security Conference and Exhibition, CSI, November 11 13,Chicago.Il.

Wong, K., Watt, S., (1990): Managing Information Security - A Non-technical Management Guide. Elsevier Science Publicers & Computer Weekly Publications, Southampton.

Wood, Michael B, (1982): Computer Security, UK.

The MobiMed Approach to Privacy in Medical Systems*

Kurt Bauknecht, Ralph Holbein, Othmar Morger,
Ulrich Nitsche, and Stephanie Teufel
University of Zurich
Department of Computer Science
Winterthurerstr. 190
CH-8057 Zurich
Switzerland
email: {*bauknecht,holbein,omorger,nitsche,teufel*}*@ifi.unizh.ch*

Abstract

We study the mutually restricting problems of permanent availability and legally demanded security of patient data in clinical information systems. Inefficiencies in today's clinical information management call for an integrated technological support for hospitals. In particluar, client/server solutions offer the flexibility and spatial independence appreciated in health care. On the other hand, mobile client applications having access from nearly everywhere in a hospital to servers containing sensitive data create severe security problems. Additionally, the need for 24 hours per day and 7 days per week availability of patient data in hospital information systems defines a conflicting border condition.

We present in this paper a proposed approach developed in the Swiss National Science Foundation project *MobiMed* to support clinical processes technologically, including access control which is compatible with security mechanisms as well as everyday practical work consideration in clinical environments.

Keywords. Mobile Clinical Information Systems, Client/Server Applications in Health Care, Security Concerns for Patient Data.

1 Introduction

Today's health care institutions, namely hospitals, lack, in most cases, efficiency from a business process point of view: patients are multiply admitted to different clinical departments of a hospital, reports and patient records are exchanged physically more than electronically across departmental borders, diagnoses are kept privately by clinical department's heads, available data is not sufficiently spread, etc. These inefficiencies call for significant changes in health care management as well as conception of health care itself. Nowadays' analyses of clinical processes as in [1] claim that hospitals have to move towards customer, i.e. patient focus in order to keep competitive in near future. To achieve such an ambitious aim in a reasonable amount of time, technological support as well as an improved information management are increasingly necessary to master upcoming challenges.

*This paper is an extended version of [10].

Technological support of hospitals' medical as well as administrative tasks reveals severe security problems that need to be solved according to legal and social demands [11]. Even today, when most data in a hospital is not publicly available in electronic form, concepts to ensure security of sensitive data are almost far from being satisfactory.

We propose in this paper a partwise solution to the discussed problems considering a client/server architecture for inter-departmental co-operation in a restricted clinical setting. Since the considered setting of a hospital prohibits radio transmission between the server and its clients, we will run a so called plug-to-plug mobility meaning that "you simply have to plug your client machine somewhere to the network to have full access to patient data according to your access rights."

According to the Swiss Data Protection Law (*Bundesgesetz über den Datenschutz*) as well as national data protection laws in other countries, storage of sensitive data, such as patient data in hospitals needs to satisfy rather strict security requirements. These are in particular: privacy (data is only accessed by authorised persons and used for intended purposes), integrity (data is not corrupted), and availability (data is available when needed). In health care there exists an additional requirement, namely that patient data like a diagnosis must not be changed after a certain amount of time.

Since the mentioned security requirements demand a wide range of measures for their fulfilment, we have to concentrate on one of them. Hence, security concerns are considered from the access control point of view in this paper: *who* is *when* allowed to access *what kind* of information? Since clinical departments show a clear hierarchical structure it is easy to identify different roles which people play in the clinical setting. Based on these roles it is, in principle, not too complicated to define *who* may access *what kind* of information. The question of *when* it is necessary for whom to get particular information is, in contrast, more difficult to handle. We discuss, therefore, in this paper an access control mechanism which is called *need to know* access control. Since immediate availability of clinical data can, in general, be life-saving in case of emergency, access control mechanisms have to be compatible to this conflicting requirement.

Technological support for the inter-departmental co-operation in hospitals taking into account security concerns of this application area is the focus of the Swiss National Science Foundation granted project *MobiMed* [2] to which the results of this paper contribute.[1] Indeed, the present paper discusses the conceptual framework for technological clinical support as was developed in *MobiMed* which is currently in its implementation phase.

After an introduction to the clinical framework for which we are currently constructing a prototype solution to support daily work, we present the technical framework which we are using for the prototype's construction: a workflow based client/server application which runs an SQL database in the background. Afterwards we discuss typical security requirements in health care and propose a solution for the access control problem. Finally, we discuss the presented solution according to its practicality in daily clinical use.

[1] *MobiMed* (Privacy and Efficiency in *Mob*ile *Med*ical Systems) is a project of the Swiss Priority Programme (SPP) *Information and Communications Structures* (ICS; 1996–1999) of the Swiss National Science Foundation (SNSF) aiming at the development of mobile access to data in a clinical environment. Its contributing members are the Computer Science Department of the University of Zurich, the Gastroenterological Department of the University Hospital of Basle, Triangle Micro Research AG, Liestal, and Basics AG, Zurich.

2 The Clinical Setting

Today's hospitals visibly are forced to become competitive in the *health care market* and are consequently driven towards *customer focus* as called in usual business contexts. Patients are considered as a hospital's customers — indeed they *are* the customers — and medical care has to focus on (Figure 1) them taking into account: cost effectiveness, satisfactory medical treatment, information transparency, time effectiveness, security of medical and personal information, comfort, etc. We view the technological support that we present in the next section as a step towards achieving patient satisfaction.

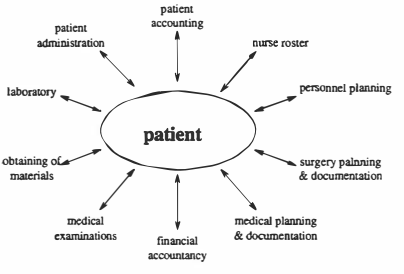


Figure 1: Today's hospitals ought to focus on their patients.

We consider two different clinical departments (investigation department, gastroenterology), while concentrating on one of them (gastroenterology) which, additionally, has a frequently used interface to another department (pathology). This framework is presented schematically in Figure 2.

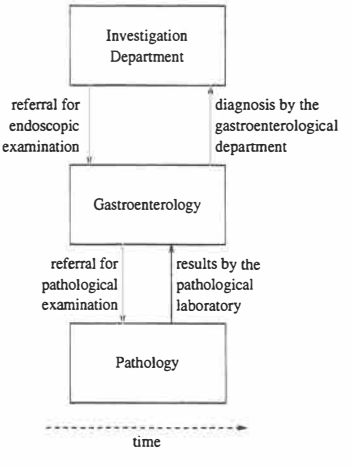


Figure 2: A typical inter-departmental co-operation.

Patients come to the investigation department, for instance, if they suffer from an unidentified illness. For further examination they may be sent to the gastroenterology

to exclude, for example, a severe stomach disease. At the gastroenterology department, endoscopic examinations take place which lead to information being added to the patient record, such as endoscopic images. To prove a particular diagnosis, tissue specimen may be removed and sent to the pathology, leading to additional entries in the patient record.

Today, the concrete setting that we consider, lacks technologically supported interfaces between the involved departments. There exists a clinical information system which only keeps personal but no medical data about patients. Patient records are kept physically rather than electronically. Consequently, diagnoses, images, information for cashing up, etc. all have to be exchanged physically, i.e. in printed form, leading to time-consuming costs-producing inefficiencies, such as retyping of patient data.

We are going to design a framework for inter-departmental co-operation in hospitals which considers and optimizes cost-effective processes. The conceptual work is accompanied by the development of an integrated prototype solution for the gastroenterology department and its co-working departments to replace its existing isolated system. Our solution aims at achieving patient focus rather than treatment focus which is today's most commonly run focus.

3 Client/Server-Based Support of Departmentwide Clinical Co-Operation

We propose a workflow-based client/server solution for the technological support of nursing as well as medical and administrative duties in health care environments. The server itself contains information on both, the patient record as well as the current state of the patient's examination and treatment which we would like to call a *health care process*. Each state in a health care process determines *who* has to perform *what* next. The server consists of an SQL database with an add-on workflow engine which keeps track and controls health care processes. Medical data is stored directly in the database; process information is stored there via the workflow engine. Control is given completely to the workflow engine which may trigger additional programs to collect or update data in a patient's record. The server is presented briefly in Figure 3. The server also handles access rights of users.

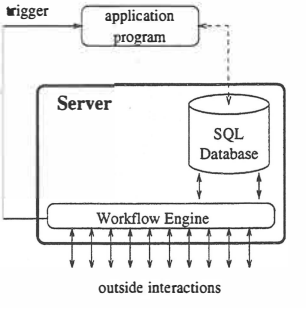


Figure 3: The server's basic architecture.

Clients are, in a simplified picture, terminal applications to access information about health care processes. An user logs in and, depending on his or her role in the hospital's

hierarchy as well as involvement in certain health care processes, receives information about his or her pending duties, may enter information according to his or her current tasks, and may access information necessary to do his or her actual work. Physically, in accordance to developments in the EU project EXODUS [3], clients will run on a personal computer that is fixed to a hospital bed. Hospital beds plus their PC units which together are called *hospital bed units* (HBU) are presented in brief in Figure 4.

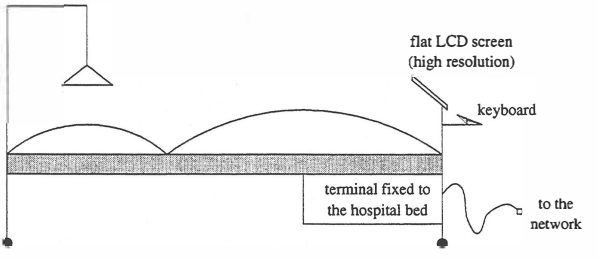


Figure 4: The hospital bed unit (HBU).

Since radio transmission is prohibited in the clinical setting to avoid the case of interaction with other medical equipment, we can only implement *plug-to-plug* mobility, i.e. the HBU's functionality is independent of the fact where it is plugged to the physical network. As a consequence of this demand, clients are not always connected to the server. To be able to handle disconnection of clients even when patient data is collected during an examination, an HBU also needs to perform restricted server functionality for, at least, keeping and adding a not too limited amount of patient data during the disconnected phase. Therefore, what is implemented on an HBU schematically looks like Figure 5. The client application which also runs a restricted server functionality is sometimes called a *fat client.*

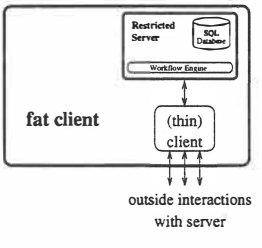


Figure 5: The fat client running on the HBU.

If the fat client is reconnected to the hospital's network and has collected data since its last connection, then it will automatically establish a connection to the server, updating information on the server according to the data kept on its local (restricted) server. This situation is sketched in Figure 6.

The server's database is intended to keep complete patient records as well as health care process information. If, for instance, the patient needs to be examined further and has to be referred to another clinical department, according to the state of the health care

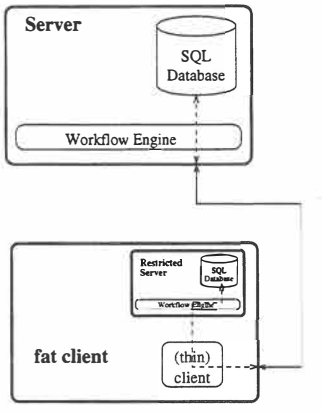


Figure 6: Updating of the server's database.

process, control is passed to other department's staff who immediately gets all necessary information about the patient's state of health to plan further examinations according to the patient's specific demands. After finishing the examination, control is given back to the department asking for further examination, where all examination data is immediately available to proceed further. After a health care process is completed, the server contains all information about the process which can afterwards be used, e.g. to make a final diagnosis, consult an expert system, or to calculate the final cashing up.

4 The Acces Control Solution

4.1 Security Requirements in Health Care

The server as described in the previous section aggregates a large amount of highly sensitive data and information respectively. Consequently, according to data protection laws (we consider in particular the Swiss *Bundesgesetz über den Datenschutz*), data kept on the server needs to be protected. There exists a variety of security requirements that have to be achieved according to legal demands. In this paper, we focus on the following requirements to be addressed by our access control system:

- Privacy: aims at protecting data against unintended and unnecessary use. For example, administrative officers, in general, do not need access to medical patient data and thus access to medical data needs to be protected against such a kind of access.
- In the medical framework, data such as diagnoses can time out in such a way that changing data that is older than a legally defined amount of time, must not be changed anymore. Consequently, write access to such a type of data has to be prohibited.

With our solution, we have to keep in mind the conflicting problem of not to hinder the availability of information by overdone access control mechanisms. We also have

to keep the balance of legal demands concerning the security of personal data and work consideration according to the application area we focus. This problem of high practicality enforces permanent evaluation of whether a developed security concept fits the matters of daily life in a hospital. In accordance to such considerations, we now present a *need to know* access control mechanism which we will use with the presented client/server architecture. The need to know access control establishes a *dynamic* role based access control mechanism by taking up a business process point of view [4].

4.2 Comprehensive Need-to-Know Access Controls

As already sketched previously, role based security approaches fit well the hierarchically structured setting of a hospital. Each level in the hierarchy corresponds to a role that people at this hierarchical level play in the hospital. After an analysis of each role's demands on obtaining particular data to perform work, access rights are assigned to each role. The access rights determine which records in the database may be read or written by a person playing a particular role. When logging in, a person identifies herself or himself by, for instance, using a chipcard and a PIN, and then a role is assigned to the person according to user/role lists, determining her or his access rights. A different way to handle role assignment is to store role information on the chipcard itself in ciphered form which then is read during log in or, alternatively, whenever data records are accessed.

Simple role based access control mechanisms have the advantage of being implementable rather easily but the drawback of certain inflexibilities. A more sophisticated access control technique refining the role based approach takes into account the question of "is it reasonable that a person playing a particluar role *needs* access to certain data at the current state of a health care process?" Obviously, it is not necessary to have access to all data about a particular patient all the time. The question of "what does one *need to know*?" gave the considered principle its name: the *need to know* principle. Implementation of a need to know principle presupposes very careful analysis of health care processes in a hospital, since access to specific data is allowed only if the system judges the access being reasonable in the particular stage of a health care process. If a particular circumstance was overseen, permission to access necessary data could be denied.

From a generic point of view the conceptual overview of our comprehensive need-to-know access control system consists of two parts as presented in Figure 7: First, there is a security design subsystem that is connected to the access control system for administration of access rights according to a need-to-know policy. The security design system includes a number of interfaces that allow to import business models, e.g. business process models (BPM), to automatically derive security information and to establish need-to-know security models [6, 7]. These need-to-know security models can be exported to a number of access control systems. Again, there are interfaces that translate the security model to specific access control information that can be interpreted by the target system. Second, there is a context authentication subsystem which can be linked to different access control systems (ACS) on different platforms via corresponding ACS client modules. These modules are hooked up within the access control systems' procedure for evaluation of access requests and allow to verify if a current need-to-know concerning a particular access request exists[5]. For that purpose it includes multiple server interfaces for different business automation systems (e.g. workflow management systems) with the capability to provide

information concerning the state of an organisation's current business transactions.[2]

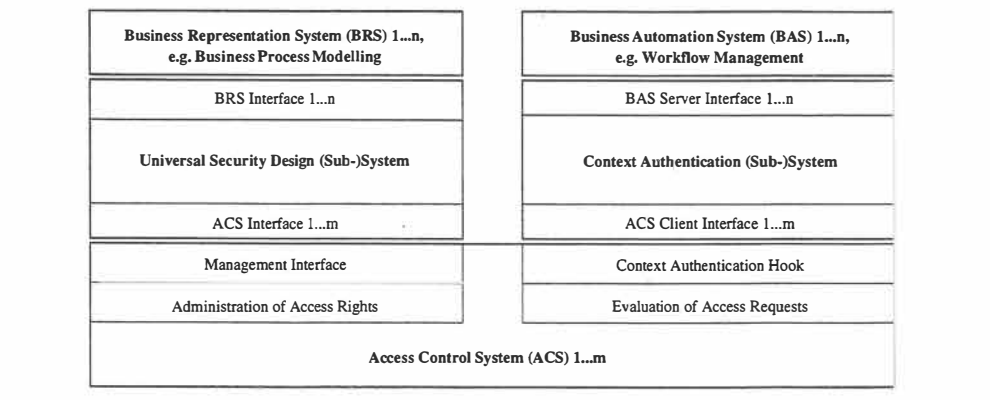


Figure 7: Conceptual overview of the access control system.

We consider now an example of an examination cycle to illustrate the need to know access control. Consider the case of a patient, after already being examined, not finding any incidents of an illness, still suffers from some pain, is referred once again to some clinical department for a re-examination. In this context it must be ensured that no information in the patient record obtained by the first examination can be corrupted, for instance, to cover up that some incidents have been overseen in the first examination. Since access control is handled depending on the current state of a health care process, we can achieve the discussed security requirement by using need to know access control: we can distinguish between already existing and new to establish entries to the patient record, by granting read only access rights to the former and read and write access rights to the latter.

To be sufficiently cautious in order to handle possible design errors for the need to know access control, one would have to implement an emergency mechanism to bypass the need to know principle, for instance by running a simple role based access control mechanism. In such a case, bypassing the need to know mechanism must be recorded carefully in a non-erasable log file to keep potential misuse of the bypass mechanism ascertainable.

5 Conclusions

We pointed out, in the present paper, that health care institutions, namely hospitals, can benefit from the use of information technology to support their health care processes. A client/server application that uses a workflow engine[3] to implement health care processes including inter-departmental co-operation appears to be a good solution for the considered specific demands. The workflow engine can also be used to implement the need to know

[2]In general, it is very useful when BRS and BAS (see Figure 7) are interrelated for example, workflow management systems that usually include a business process modelling component as well as a workflow control engine. However, this is not necessary in principle.

[3]The runtime environment of a workflow system.

access control mechanism [4, 8].

Security concerns in health care are discussed on two different levels: the security demands in hospitals are discussed and afterwards solutions to the access control problem are sketched. A role based access control mechanism to patient data records in an underlying SQL database offers a quite easy to implement solution to control data accesses. A more sophisticated need to know access control principle offers a finer access control technique than the simple one.

The discussed client/server application for the support of health care processes is currently in its development and implementation phase, embedded in the Swiss National Science Foundation project *MobiMed* in co-operation with the EU supported project *EXODUS* [3]. *MobiMed* aims at developing a prototype solution for the inter-departmental co-operation at the University Hospital of Basle. The prototype will run on an *Microsoft SQL* database with an add-on *ActionWorkflow* specification and run-time module [9] (see Figure 8) to implement health care processes. For states in a health care process that offer or demand direct access to the SQL database, additional software triggered by the workflow engine is to be implemented.

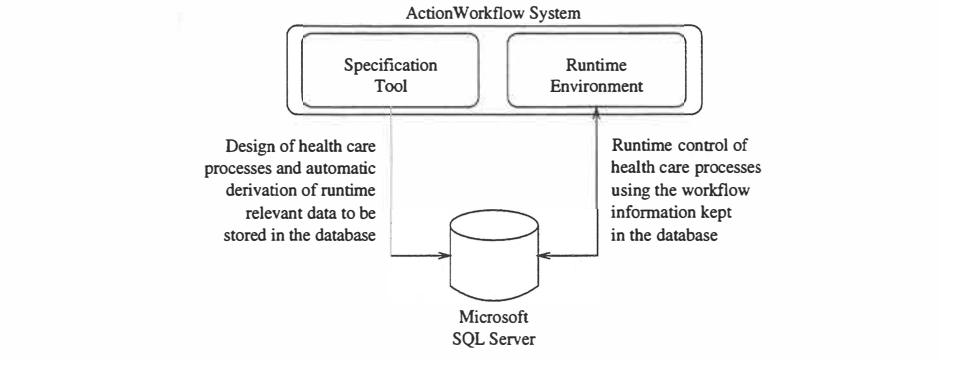


Figure 8: The software-components used to control our system.

As already mentioned, *MobiMed*, having completed its conceptual phase, is currently in its implementation phase. Further work will be: adaption to user requested requirements, evaluation of the economical as well as process oriented benefits of the prototype in clinical daily life, and finally, considering demands for a complete hospital-wide technological support based on the results obtained in *MobiMed*, including administration, obtaining of materials, pharmacy, etc.

References

[1] M. Brucksch and H. Grabowski. Auf dem Weg zur Hochleistungsklinik — Von der Planwirtschaft zur wettbewerbsfähigen Patientenversorgung. In A. Little, editor, *Management in vernetzten Unternehmen*, pages 259–283, Wiesbaden, 1996. Gabler Verlag.

[2] H.-R. Fischer, S. Teufel, C. Muggli, and M. Bichsel. Privacy and Efficiency of Mobile Medical Systems (MobiMed). Technical report, Bewilligtes Forschungsgesuch des Schwerpunktprogramms Informatikforschung SPP–IuK, Modul: Demonstrator, Nr. 5003-045359, 1995.

[3] J. Francis, D. Polymeros, G. Lyberopoulos, V. Van de Keere, A. Elberse, P. Rogl, and L. Vezzoli. Project EXODUS: Experimental mobile multimedia deployment in atm networks. In *Submitted to the Journal of the ACM*, 1997.

[4] R. Holbein. *Secure Information Exchange in Organizations.* PhD thesis, University of Zurich, Switzerland, 1996. published by Shaker Verlag, Aachen.

[5] R. Holbein and S. Teufel. A context authentication service for role based access control in distributed systems – CARDS. In J. Eloff and S. von Solms, editors, *Information Security – the Next Decade IFIP/SEC'95.* Chapman & Hall, London, UK, 1995.

[6] R. Holbein, S. Teufel, and K. Bauknecht. A formal security design approach for information exchange in organisations. In D. Sponner, S. Demurjian, and J. Dobson, editors, *Proc. 9th IFIP TC11 Annual Working Conference on Database Security 1995.* Chapman & Hall, London, UK, 1996.

[7] R. Holbein, S. Teufel, and K. Bauknecht. The use of business process models for security design in organisations. In S. Katsikas and D. Gritzalis, editors, *Information Systems Security – Facing the Information Society of the 21st Century IFIP/SEC'96.* Chapman & Hall, London, UK, 1996.

[8] R. Holbein, S. Teufel, O. Morger, and K. Bauknecht. A comprehensive need-to-know access control system and its application for medical information systems. In L. Yngström and J. Carlsen, editors, *Information Security in Research and Business IFIP/SEC'97.* Chapman & Hall, London, UK, 1997.

[9] R. Medina-Mora, T. Winograd, R. Flores, and F. Flores. The action workflow approach to workflow management technology. In *Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW'92)*, pages 281–288, Toronto, 1992. ACM Press.

[10] O. Morger, U. Nitsche, and S. Teufel. Security concerns for mobile information systems in health care. In *Proceedings of the 8th International Workshop on Database and Expert Systems Applications (DEXA'97)*, Toulouse, France, 1997. IEEE Press.

[11] S. Teufel and R. Holbein. Security aspects of mobile medical systems – a project overview. In J. Eloff, editor, *Proceedings of the IFIP TCII WG 11.2 Small System Security*, Samos, 1996.

Fast Payment Scheme for Electronic Fee Collection

Cristian Radu[1], Rabah Boussouira[2]

[1] Katholieke Universiteit Leuven, Laboratorium ESAT
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
email: Cristian.Radu@esat.kuleuven.ac.be
[2] Advantec Ltd
Lauttasaarentie 44A1, FIN0200 Helsinki, Finland
email: rbo@advantec.fi

Abstract

This paper outlines the design of a payment scheme that can be used for Electronic Fee Collection (EFC) on the highways. This application shows a high market potential with respect to electronic payments. The major functional requirements of the user in connection with the EFC payment scheme are that the payment transaction should be reliable and fast enough, such that the user can pass the toll station at a high speed. Our solution fulfills these functional requirements and proposes the security measures that are necessary in order to protect the integrity of all the parties involved in the EFC system. Both the communication loading and the computational effort are reduced during the payment transaction. The algorithms specifying the payment scheme can be easily implemented using 8-bit microprocessor based smart cards.

1 Introduction

An *electronic payment scheme* consists of a set of participants and their transactions towards an efficient exchange of values among them. The *transactions* include the complete process of issuing and use of electronic payment instruments as well as the clearing and settlement of financial operations. In an extended scenario, as shown in Figure 1, the *participants* can be considered in connection with five essential roles: the *issuer*, the *user (payer)*, the *service provider (payee)*, the *collection agent* and the *clearing operator*.

Both the set of participants and transactions are determined to a large extent by the specific payment method which is adopted at the core of the scheme. The *payment method* is determined by three dimensions: the electronic payment instrument (also called the payment means), the payment mode and the payment scope [11].

The *electronic payment instrument* represents the expression of an a priori agreement between the issuer and the user on the possibilities of the user to access the services available in that payment system. Some of the most common examples of

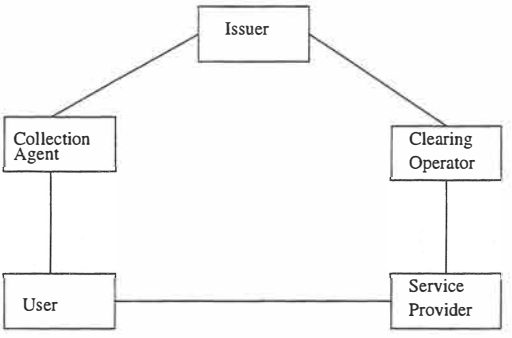


Figure 1: The general model of the electronic fee collection scheme

electronic payment instruments are: electronic cash [1, 4, 5, 6], electronic cheques [3], electronic debit instruments [2, 7, 13] and electronic credit instruments [14]. A *payment claim* specifies the details of a payment transaction: the amount to be paid, the currency in which this amount is expressed, the time and date when the transaction was carried out, the identity of the service provider to whom the payment is intended, and a serial number or a random number that identifies the payment transaction in a unique way. The electronic payment instrument is used to produce *evidence* on a payment claim forwarded by the service provider during a payment transaction. The evidence consists of a transformation of the payment claim, parameterized by some information that is specific to the electronic payment instrument (see Figure 2).

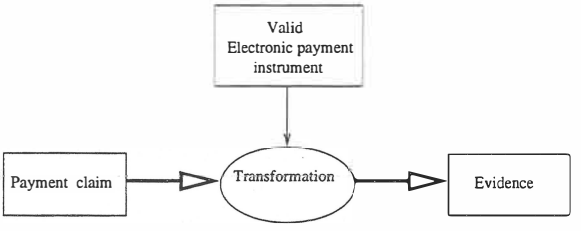


Figure 2: The generation of an evidence on a payment claim

The *payment mode* is the parameter defining the time when the value of the payment instrument is deducted from the user's money, e.g., pre-payment or post-payment.

The *payment scope* is the application extension of a payment method. *Electronic Fee Collection* (EFC) provides a promising tool for the large scale introduction of toll collection. EFC enables users of road infrastructure to pass through toll stations without stopping, but requires user equipment for payment of fees. The EFC scope

concerns those transport services which will have to be accompanied by a payment. Some examples of services are: highway tolling, area access control, parking booking and charging, public transport, charging for the transfer of traffic information to and from the vehicle. However, the payment scope of the EFC payment scheme, as described in this paper, is limited to highway tolling.

The remainder of the paper is organized as follows. Section 2 presents the functional and security requirements of the roles that are participating in the system. Section 3 describes which is the equipment that is needed by the user and the service provider in order to accomplish the functionality of the EFC system. Section 4 details the main transactions of the EFC payment scheme, namely the withdrawal of payment instruments, the payment and the deposit of payment transcripts. This design solution is one of the alternatives proposed in the framework of the European Union's project **MOVE**-*it*: Motorway Operators Validate EFC for Interoperable Transport [11, 12]. Finally, our conclusions are presented.

2 Design requirements

The design of the EFC payment scheme is carried out in a simplified scenario, where the role of the issuer is played by a bank, and the role of the service provider is assumed by a single motorway operator. Furthermore, the service provider and the users are clients of the same bank (the issuer). In this simplified scheme, the clearing operator and the collection agent are not required, and are therefore skipped. However, the scheme can be adapted to support a multi-issuer/multi-service provider environment. The simplified EFC scenario is presented in Figure 3.

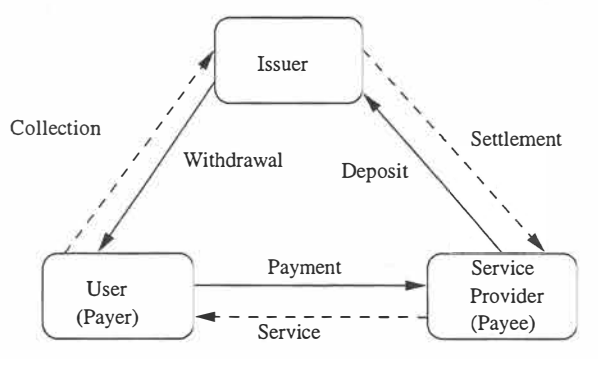


Figure 3: The simplified EFC scenario

2.1 Functional requirements

The major functional requirements of the user in connection with the EFC payment scheme are that the payment transaction should be reliable and fast enough, such that the user can pass the toll station at a high speed.

There are two possibilities to collect fees for the highway related services in the EFC systems:

- *open system*: all vehicles or users belonging to the same category pay the same fee when passing a given gate (toll station); and
- *closed system*: a user of a given category pays the fee for the travel she has made when she leaves the highway network. The fee is computed according to the distance traveled inside the network.

The functional requirement of the service provider is that the payment scheme should be used in an open EFC system. Therefore, what is paid is the access to the infrastructure and not the actual distance that is traveled.

The payment mode adopted in the fast EFC payment scheme is characterized as follows:

- The user has a local account, the balance of which is upper bounded by a maximum value max_bal. This parameter is agreed between the user and the issuer, when the former sets up a contract with the latter. The local account is kept in the electronic purse of the user. The secure management of the local account requires that the purse of the user must be implemented by a smart card, which is assumed to be a tamper-resistant device.
- The issuer keeps a shadow account, corresponding to the local account of the user.
- The scheme is debit-based, in the pre-payment/off-line mode. In a *pre-paid* payment scheme the balance of the user's local account is credited with an initial amount *before* a service is actually provided. If there exists a shadow account, its balance is increased with the initial amount of money which the user pays to the issuer during the collection transaction. From this initial amount, the user can later load smaller amounts in her local account, by increasing its balance with the exact amount deducted from the shadow account. In an *off-line* payment scheme the user can use her electronic payment instruments in order to pay an amount to any service provider without to establish a communication with a third party (e.g. the issuer) during the payment transaction. The service provider forwards to the issuer the transcripts collected during a session of payment transactions, after these transactions took place.
- The service provider charges the user separately for each passing of the toll station – *payment per event mode.*

In every payment transaction the user produces an evidence on the payment claim of the service provider. The evidence consists of a message authentication code (MAC) produced with respect to a *session secret key* SSK (see Figure 4). This session key is derived from the *debit secret key* DSK, which is generated by the issuer during the personalization of the payment instrument, and is uniquely linked to a user. The debit secret key represents the *electronic debit instrument* of the user.

Thus, the fast EFC payment scheme can be characterized as off-line, local account, pre-payment, payment per event, using electronic debit instruments.

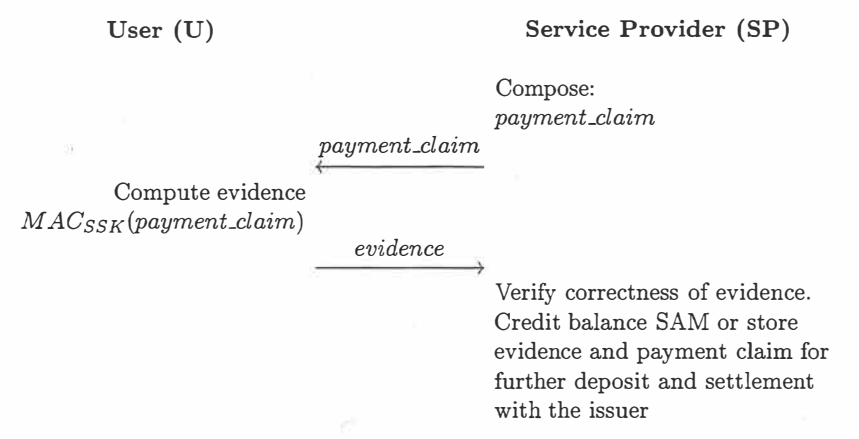


Figure 4: Overview of the payment transaction

2.2 Security requirements

The honest user – who pays for transportation services and respects her duties in the EFC system – needs to be assured that the following integrity requirements are fulfilled:

- The user needs to authenticate the issuer during the collection and withdrawal transactions and the service provider during the payment transaction. The user must be able to verify that the issuer and the service provider and all equipment used by them are legitimate.
- Whenever the user credits an amount in her shadow account kept with the issuer, the balance of this account must be increased accordingly. The balance of the shadow account is decreased with the exact amount required by the user to be transferred in her local account.
- During the collection transaction, the balance of the local account is increased with the amount acknowledged by the user to be transferred from her shadow account. The balance of the local account is decreased with the exact amount specified in a payment claim by a legitimate service provider.
- Whenever the user respects the contract with the issuer, she is entitled to obtain the electronic debit instrument used to produce evidence in the payment transaction. The debit instrument must be validated by the issuer, in such a way that the evidence produced by the user on the payment claim to be acceptable at the deposit stage (with the issuer).
- In the specific cryptographic implementation of the fast EFC payment scheme, the debit secret key DSK of the user is known to the issuer. However, the framing by the bank attack is not likely to appear, otherwise the user would lose her trust in the bank. Therefore, it is acceptable to assume that during the payment transaction, no electronic debit instrument can be used to

produce evidence on a payment claim on behalf of the user without the cooperation of her SC. It is also assumed that only the legitimate user can transfer money from her shadow account to the corresponding local account, during a collection transaction carried out with the issuer.

The service provider is interested in the fulfillment of the following requirements:

- Both the user and her equipment are legitimate in the EFC system.
- The service provider has a local account managed in a secure way through a tamper-resistant Secure Access Module (SAM). The issuer provides the service provider with a SAM at the moment when the service provider joins the EFC system and opens an account with the issuer. The balance of the SAM's local account is increased with the amount specified by the service provider in the payment claim, for which evidence is produced using a valid electronic debit instrument.
- If a user produces evidence on a payment claim, using to this end a valid electronic debit instrument, later on, the service provider can deposit the corresponding payment transcript with the issuer and he can recover his money.
- The amount of money specified by the balance of the SAM's local account for which there is evidence produced by the users, can only be credited to the account indicated by the service provider.
- The service provider cannot falsely be accused by the issuer of double depositing payment transcripts.

The security of the issuer is provided if the following integrity requirements are fulfilled:

- The electronic debit instrument cannot be forged. The issuer will accept the settlement of those payment transcripts for which evidence was produced by one of its clients, using a valid electronic debit instrument. This instrument can be obtained only by the user that adopted the payment method of the issuer.
- The issuer will accept to refund the service provider for each payment transcript where evidence was produced by one of the users and which was not previously paid. To this end, the issuer manages a database in connection with each user. This database records all the payment transcripts accepted by the issuer in connection with the user, which were paid to the service provider during the validity period of the user's instrument.
- During the collection transaction, the issuer is the only one entitled to increase the balance of the local account of the user, mirrored by the corresponding decrease of her shadow account.

3 Specific equipment for EFC

At the physical level, the user and the service provider are represented by the *On-Board Equipment* (OBE) and the *Ground Equipment* (GE), respectively (see Figure 5).

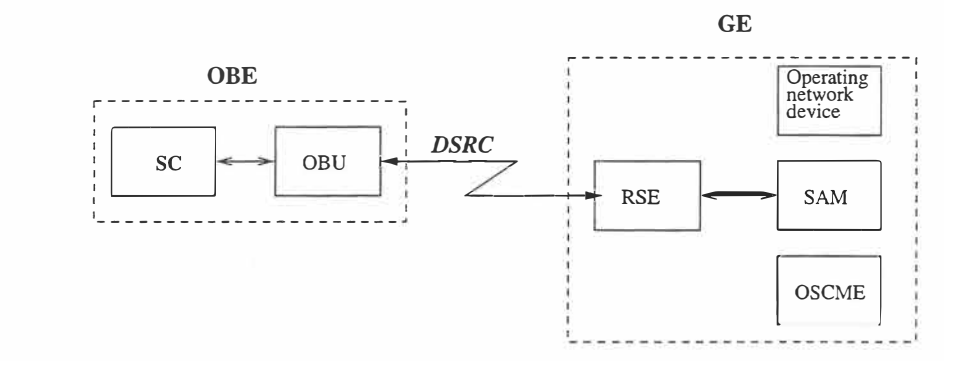


Figure 5: Specific equipment for the user and service provider in the EFC system

The OBE is installed in the vehicle of the user. It is designed to ensure transmission and reception of data going to and coming from the Ground Equipment according with the Dedicated Short Range Communication ($DSRC$) standard, to store data related to the electronic debit instrument, and to manage interactions with the user. When a pre-payment mode with local account is adopted, the OBE is implemented by a combination of two devices: the *On-Board Unit* (OBU) and the *smart card* (SC). The functionality of these two devices can be briefly described as follows:

- The main function of the OBU is to manage the $DSRC$ link between the OBE and the GE. The OBU can also provide the interface with the user, when this is required. The OBU is transparent with respect to the protocols of the transactions that describe the payment scheme.
- The SC is a tamper-resistant device that keeps the local account of the user and her electronic debit instrument.

The Ground Equipment (GE) implements the functionality of a toll station within the EFC system from the point of view of the service provider. The GE performs in real-time the functions related to the $DSRC$ link with the OBE of the user. The goal of this link is to fully support all the phases involved by a payment transaction between the user and a service provider. The device within the GE that carries out this task is referred to as the *Road Side Equipment* (RSE) [8]. The local account of the service provider is managed in a secure way through the use of a Secure Access Module (SAM).

The GE is also responsible for the network communication established between the service provider and the issuer during the deposit transaction. The corresponding device is referred to as the *Operating Network* (device).

The *Off-line Smart Card Management Equipment* ($OSCME$) is an optional subsystem of the GE. It can provide operational support for the user willing to read and/or update her SC: read the balance of the local account and load the local account of the user with a certain amount, through a connection to her shadow account kept with the issuer.

4 Fast EFC payment scheme

The scheme uses a centralized approach, where the issuer trusts tamper-resistant devices:

- Secure Access Modules (SAM) in the GE of service providers. The SAM stores the keys that are necessary during the protocols and the local account of a service provider.
- Smart Cards (SC) in the OBE of users. The SC stores the electronic debit instrument, the keys that are necessary during the protocols, and the local account of the user.

In the remainder of this section we first detail the initialization stage of the EFC payment scheme. Afterwards, we concentrate on the payment protocol, carried out between the OBE of the user and the RSE of the service provider. The collection, withdrawal, and deposit transactions are not detailed in this paper. However, we assume that the balance of the local account of the user was increased through a collection transaction and that the user has a valid electronic debit instrument. This is obtained for the first time during the registration of the user and it can be refreshed through a withdrawal transaction any time it expires.

4.1 Setup of the EFC payment scheme

In the case when the same *master key for debit* MKD is stored in all the SAMs, the overall security of the system is highly dependent on the tamper-resistance of every SAM. If the tamper-resistance of *one* SAM is compromised, the attacker can take control over the whole system. Therefore, it is preferable to avoid the solution where all the SAMs store the same master key for debit, such that if the tamper-resistance of a SAM is broken, the damage is reduced to the group of SAMs sharing the same master key.

To this end, all the SAMs in the system are divided in G groups, each group containing M_g, $g = 1, \ldots, G$ units. Each group is characterized by a different master key for debit MKD_g, $g = 1, \ldots, G$, which is stored in each SAM_{gi}, $i = 1, \ldots, M_g$ of the group during its manufacturing. If the tamper-resistance of SAM_{gi} is successfully broken, then the master key for debit MKD_g is obtained and the attacker can simulate any SAM in the group g. However, the security increases

since the attacker has to be able to break the tamper-resistance of one SAM from each group $g = 1, \ldots, G$ in order to be able to take control over the whole system.

Since the user can attempt to break the tamper-resistance of a SC, a basic caution requires the diversification of keys in the SCs. Each SC, SC_j, $j = 1, \ldots, N$, stores a separate diversified debit secret key DSK_{gj} with respect to each group $g = 1, \ldots, G$ of SAMs in the system.

The diversified debit secret key DSK_{gj} is obtained by computing a MAC with respect to the master key for debit MKD_g on the concatenation of the identifier of the issuer ID_I and the serial number SC_j_snum of the smart card SC_j (see Figure 6).

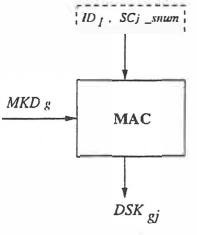


Figure 6: The generation of diversified debit secret keys in the SC

For every group $g = 1, \ldots, G$ of SAMs existing in the system, and for every smart card SC_j, $j = 1, \ldots, N$ the issuer computes the diversified debit secret key DSK_{gj}. All the keys DSK_{gj}, $g = 1, \ldots, G$ are stored in the smart card SC_j during the registration of the user with the issuer.

Each SAM_{gi}, $g = 1, \ldots, G, i = 1, \ldots, M_g$ can re-compute the key DSK_{gj} during the payment transaction carried out with any of the smart cards SC_j, $j = 1, \ldots, N$, based on the data ID_I, SC_j_snum received from the OBE of the user, using the same procedure like that described in Figure 6. This is possible because the SAM_{gi} has stored the master key for debit MKD_g since the manufacturing stage. The distribution and generation of keys is summarized in Table 1.

	DSK_{gj}	MKD_g
SAM_{gi}	computed during each payment transaction	stored in the SAM_{gi} since the manufacturing stage
SC_j	DSK_{gj} generated at registration by the issuer	None

Table 1: The generation and distribution of keys

The map of the keys stored in SAMs and SCs is schematized in Figure 7.

The generation of the session secret key SSK for the payment transaction is derived by computing a MAC with respect to the diversified debit secret key DSK_{gj}

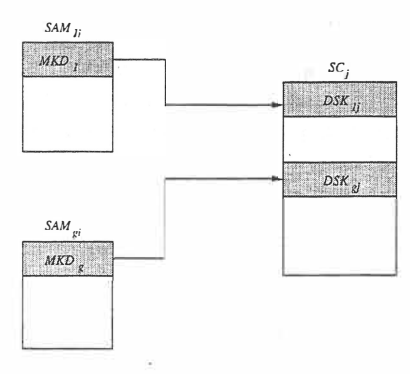


Figure 7: The map of keys in SAMs and SCs

on the concatenation of the serial number nt_SC of the transaction in the SC and the serial number nt_SAM of the transaction in the SAM (see Figure 8).

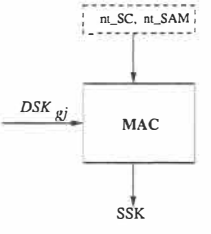


Figure 8: The generation of the session secret key SSK for payment

The message authentication code (MAC) that is used can be a CBC-MAC derived from a block cipher [9, 10].

The internal representation of the electronic debit instrument of the smart card SC_j consists of the set of diversified debit secret keys DSK_{gj}, $g = 1, \ldots, G$. The external representation of the electronic debit instrument of the smart card SC_j consists of the items SC_j_snum and ID_I from which any SAM_{gi} can compute the corresponding diversified debit secret key DSK_{gj}.

4.2 Payment protocol

The critical protocol from the point of view of time is that implementing the payment transaction. The step by step description of this protocol is the following [12]:

Step 0 During the *initialization phase*, a communication channel is established between the RSE of the service provider and the OBE of the user that has entered the $DSRC$ zone of the open EFC system, but has not yet established a communication with the RSE.

Step 1 During the *presentation request phase*, the RSE sends the message M_1, which contains details about the EFC transaction point: beacon location, beacon type, the identifier ID_{SP} of the service provider as well as the identifier of the SAM in the RSE, denoted ID_{SAM}, and the group g to which the SAM belongs.

Step 2 In the *presentation response phase*, the OBE provides the RSE with information about the vehicle itself and the electronic debit instrument that is going to be used in the transaction result phase (Step 4). To this end, the OBE composes the message M_2 from the following items:

- M_{21} = the category of the vehicle;
- M_{22} = the external representation of the electronic debit instrument of the user DI_u:
$$DI_u = ID_I, SC_j\_snum;$$
- M_{23} = the current number of transaction nt_SC executed in the SC.

The OBE (on behalf of the user) sends $M_2 \leftarrow M_{21}, M_{22}, M_{23}$ to the RSE. The message M_2 must contain all the necessary information that allows the RSE (on behalf of the service provider) to decide if the OBE can use the EFC facility in order to pay and what amount has to be paid.

Step 3 In the *transaction request phase*, the RSE receives the message M_2. The SAM computes the diversified debit secret key DSK_{gj} corresponding to the smart card SC_j in connection with the group g of SAMs, based on the external representation of the electronic debit instrument of the user $M_{22} = DI_u = ID_I, SC_j\_snum$:
$$DSK_{gj} = MAC_{MKD_g}(DI_u).$$
Using the key DSK_{gj} computed by the SAM, the RSE computes the session secret key corresponding to current payment transaction as:
$$SSK = MAC_{DSK_{gj}}(nt\_SC, nt\_SAM)$$
The RSE also compares the category of the vehicle declared by the user (message M_{21}) with the category of the vehicle measured at the gantry. If the vehicle category was correctly declared, the RSE informs the OBE on whether it is authorized, and if so, how much it has to pay. The amount to be charged (ch_amount) can be calculated by the RSE based on the information contained in the message M_{21} forwarded by the OBE in the presentation response phase (Step 2). In this computation there are involved other parameters as well, like for example the time of the day when the highway is used. The RSE forms the message M_3 from the following items:

- M_{31} = "Yes" or "No" acknowledge about the authorization to use the EFC facility (the user is authorized to use the EFC system if the declared vehicle category is correct);
- M_{32} = the amount to be charged as a fee (ch_amount);

- M_{33} = the time of the day and the date when the transaction takes place (time, date);
- M_{34} = the current number of the transaction nt_SAM carried out by the SAM.

The RSE also computes the access credentials that entitles the SC to debit the balance of the local account with the amount specified in the transaction:

$$access\_credentials \leftarrow MAC_{SSK}(M_{31}, M_{32}, M_{33}, M_{34}).$$

The RSE sends the message $M_3 \leftarrow M_{31}, M_{32}, M_{33}, M_{34}$ together with the $access\_credentials$ to the OBE. The SAM composes the payment claim:

$$claim \leftarrow M_1, (M_{21}, M_{23}), (M_{32}, M_{33}, M_{34}).$$

and computes the verification value $V_{val} = MAC_{SSK}(claim)$. Both $claim$ and V_{val} are stored in the non-volatile memory of the RSE.

Step 4 In the *transaction result phase*, the SC evaluates the correctness of the access credentials received from the RSE. To this end the SC first compute the session secret key SSK:

$$SSK = MAC_{DSK_{gj}}(nt\_SC, nt\_SAM)$$

and verifies that:

$$access\_credentials \stackrel{?}{=} MAC_{SSK}(nt\_SC, nt\_SAM).$$

If this verification holds true the SC decreases the balance of the local account with the amount specified by the message M_{32}. The SC also computes the evidence on the payment transaction as:

$$evidence \leftarrow MAC_{SSK}(claim).$$

The OBE sends the evidence to the RSE.

Step 5 The RSE receives the evidence produced by the OBE and sends it to the SAM. The SAM compares $evidence$ with the V_{val}. If the verification holds true, the balance of the local account in the SAM is increased with the amount specified in the claim. Otherwise, an enforcement procedure is started by the RSE against the user.

The RSE stores the electronic debit instrument M_{22}, the payment claim $claim$, and the evidence $evidence$ produced by the user on the payment claim with respect to the internal representation of the debit instrument. These three items form the *payment transcript*, which is forwarded to the issuer at the deposit stage. The correctness of the evidence $evidence$ allows the service provider to recover his money from the issuer.

5 Conclusion

The use of secret key technology allows the implementation of security measures in the EFC system. Thus, during the payment transaction the overhead introduced by the security measures consists of the computation of five message authentication codes, from which only two are computed by the smart card and the rest are computed by the *SAM*. With the current status of the art in smart card and *SAM* technologies, the time required for the implementation of security measures allows total transaction times to satisfy the performance requirements for free flow road tolling EFC. Therefore, this is a practical approach towards the implementation of a secure fast EFC payment scheme that allows collection of tolls on the highways. The scheme can be successfully adopted as a "proprietary" scheme by an issuer or a consortium of a limited number of issuers.

References

[1] S. Brands, *An Efficient Off-line Electronic Cash System Based On The Representation Problem*, Report CS-R9323, Centrum voor Wiskunde en Informatica, April 1993.

[2] Bull CP8 IEP-CC Cards. User Guide, Vol.1, Reference TU 0258 A01, December 1994.

[3] CAFE Consortium, *Final Report Volume IIA. Technical Specifications: Architecture and Protocols*, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands, to appear.

[4] D. Chaum, A. Fiat, M. Naor, "Untraceable electronic cash," *Advances in Cryptology, Proc. Crypto'88, LNCS 403*, S. Goldwasser, Ed., Springer-Verlag, 1990, pp. 319–327.

[5] http://www.digicash.com/ecash/quickref.html

[6] M. Franklin and M. Yung, *Towards provably secure efficient electronic cash*, Technical report CUCS-018-92, Columbia University, Department of Computer Science, April 1992.

[7] *Identification Card Systems – Inter-sector electronic purse. Part 2: Security Architecture*, CEN/TC224/WG10, December 1994.

[8] *Interface Specification for Clearing between Operators*, Automatic Fee Collection (AFC), RTTT-CEN/TC278/WG1, December 1995.

[9] ISO 8731:1987, *Banking – approved algorithms for message authentication, Part 1, DEA,* ISO 8731-1, *Part 2, Message Authentication Algorithm (MAA),* ISO 8731-2.

[10] ISO/IEC 9797:1993, *Information technology – Data cryptographic techniques – Data integrity mechanisms using a cryptographic check function employing a block cipher algorithm.*

[11] C. Radu, A. Bosselaers, H. Lormans, R. Boussouira, W. Gygli and W. Asmuth, *Security – Threat and Risk Analysis for the Electronic Fee Collection System*, **MOVE**-*it* Deliverable 3.1, Activity 032, Project TR 1105, EC-DG XIII, Telematics Application Programme, December 1996.

[12] C. Radu, A. Bosselaers, H. Lormans, R. Boussouira, W. Gygli and W. Asmuth, *Security Framework for Electronic Fee Collection*, **MOVE**-*it* Deliverable 3.3, Activity 032, Project TR 1105, EC-DG XIII, Telematics Application Programme, July 1997.

[13] C. Radu, R. Govaerts and J. Vandewalle, "Prepaid electronic cheques using public-key certificates," *Cryptography and Coding, Proceedings of the 5th IMA Conference, LNCS 1025*, C. Boyd, Ed., Springer-Verlag, Cirencester (UK), 1995, pp. 132–141.

[14] *Secure Electronic Transaction (SET) Specifications*, available on:
http://www.mastercard.com/set/set.htm

Building Secure High Speed Extranets

Chandana Gamage Jussipekka Leiwo Yuliang Zheng

Peninsula School of Computing and Information Technology
Monash University
McMahons Road, Frankston, Vic 3199, AUSTRALIA
Phone +61-(0)3-9904 4287, Fax +61-(0)3-9904 4124
E-mail:{chandag,skylark,yuliang}@fcit.monash.edu.au

Abstract

Extranets are a technology for creating logical views of geographically separate LANs by providing a transparent interconnection mechanism between them over a WAN. As LAN bandwidths are typically higher than those of a WAN, and LANs typically support stronger security features, it is essential that extranets can be constructed to be both secure and high of speed. The security of an extranet can be significantly reduced if the interconnection mechanism is not capable of providing comparable level of security to that of LANs being interconnected. Typically, high speed and security require trade offs, but in this paper a method shall be proposed for employing IP switching technology and a new public key cryptographic paradigm, digital signcryption, to construct extranets with both high speed and high level of security.

1 Introduction

An extranet is a private virtual network that interconnects several local subnets (or intranets) over a wide area network (WAN) to carry IP datagram [13] traffic. A typical example of an extranet is the interconnection of geographically distributed local subnets belonging to a single organization using the global Internet. The individual subnets that are part of an extranet are usually local area networks (LAN) with completely localized management and control over its operation. Another type of extranet can be constructed by interconnecting parts of intranets belonging to several organizations through the public Internet to form a shared private internet.

The main identifying attribute of an extranet that differentiates it from a LAN or a WAN is its administrative control structure. A LAN operates under centralized local control policies while a WAN operates under distributed cooperative control. In contrast, an extranet is a collection of *logical views* of private intranets (LANs) that are interconnected over the public Internet to form a private virtual network. For example, this logical view can a be portion of a local subnet partitioned through a packet filtering gateway or permission to execute only a controlled set of applications by blocking external access at host network connectivity ports. Unlike in normal LAN or WAN implementation, security is an integral design feature of extranets. Additional design features are interoperability, reliability and performance.

Interoperability concerns are addressed through standard Internet protocols and related technology on which extranets are constructed of. If the underlying network infrastructure is

solely based on IP technology, network layer interoperability is not a significant issue. However, given the heterogeneous nature of the Internet, achieving of application layer interoperability requires significant additional work. Typical approach is deployment of middleware such as CORBA/ORB and DCE to support distributed applications. On application layer, extranet provides participating organizations with a network infrastructure to integrate their interacting business processes by executing distributed applications and sharing data. Such a distributed on–line systems allow implementation of many of the models for carrying out electronic commerce in a secure, efficient and reliable manner. Example applications for extranets include electronic data interchange (EDI) based systems such as just–in–time (JIT) manufacturing and inventory control [5], work flow document management and real time funds and stock portfolio management. High performance extranets can be used for collaborative interactive work such as industrial design and modeling or real–time multimedia conferencing.

Reliability of extranets at the network layer is based on the same robust and fault–tolerant IP datagram based network model on which Internet operates at. Provision of the necessary performance for high bandwidth and delay sensitive applications together with secure operation of extranets are the areas in which new solutions are required, they have not been traditionally treated as core issues by Internet developers. Therefore, it is justified and important to study them from an extranet construction point of view. Within this paper they are studied in concert with the aim of providing a secure–integrated research approach towards construction of high speed extranets.

The paper is started by surveying models for building extranets in section 2. This survey is then followed by an outline of methods for constructing high speed extranets in section 3. Section 4 provides with an overview of an integrated IP switch firewall. Conclusions are drawn and directions highlighted for future work in section 5.

2 Security in Extranets

Extranets for low bandwidth applications can use available Internet connections for routing of IP datagrams between participating intranets over the public WAN as shown in figure 1. For extranets supporting applications with sustained high bandwidth requirements, leased communication circuits may be necessary. The connectionless datagram based IP model is a flexible and robust networking mechanism over failure prone WANs. However, in an extranet environment, it is necessary to support integrated services with diverse quality of service (QoS) requirements for bandwidth allocation, delay and throughput over this network infrastructure. For this purpose, IP model can be extended by adding a capability to cache soft states relating to the flow of packets across internetworking units such as routers and switches.

An example of this method is the reservation protocol (RSVP) that does bandwidth allocation at the network layer to increase the throughput by reducing the potential for packet loss due to buffer overflow and subsequent retransmissions [15]. The RSVP protocol mechanism is limited in its capacity to make significant improvements to the end–to–end latencies as no attempt is made to speed up the processing of individual network protocol data units (NPDUs) as they pass through an interworking unit. As the focus of this paper is on high speed extranets including high packet throughput at routers, section 3 presents a more detailed discussion on this aspect.

In extranets, security gateways or firewalls provide secure external network access to trusted hosts located internal to a local subnetwork. Multiple security gateways operating in coordi-

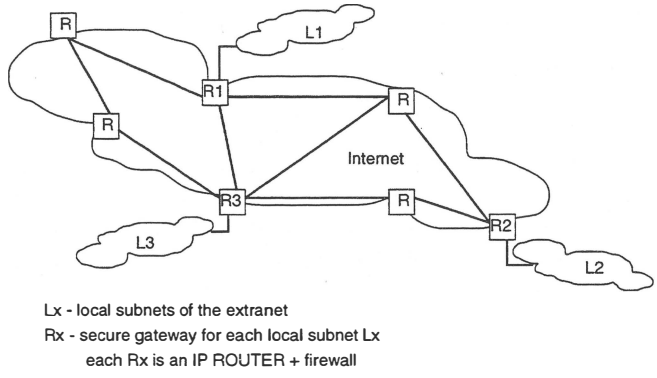


Figure 1: Building extranets using firewalls

nation are used to build the secure virtual private network for the extranet. A gateway based secure networking environment can efficiently implement a host–oriented keying scheme (as against a user–oriented keying scheme) by performing all the *network layer* security related processing at the gateway thus limiting the number of hosts involved in the key management scheme. In this mode of operation, the secure gateway functions as a key management proxy for the trusted hosts inside its subnet.

Enforcement of security can be divided into two categories: provision of security at the point of entry to a network and provision of security of transmitted data. Security at the point of entry is usually enforced by authentication and access control schemes and protection of data during transmission by authenticity, confidentiality and integrity services.

2.1 Security at the Point of Entry

A firewall is a single point of entry to a protected intranet and provide its first line of defense against *outside* attacks. However, firewalls are not transparent in their operation due to the need for close interaction with supported applications. A firewall can block or grant access based on a combination of criteria including source and destination IP addresses (by IP packet level filtering), network port connection type (by TCP/UDP frame level filtering) and application specific attributes such as user authentication credentials (by application frame level filtering) [11]. The cost of filtering related processing increases as the protocol stack layer at which we perform the function increases. However, the flexibility and range of filtering that can be carried out also increases as we go up the protocol stack giving the extranet designers with a performance/flexibility trade–off. In actual implementations, an extranet built using firewalls is likely to perform filtering at each level of the protocol stack. The main disadvantage of firewall technology is that it is modeled on perimeter security from external attacks, therefore limiting its protection capabilities against internal attacks.

Access control systems play a major role in secure extranets. While a firewall is tasked with prevention of unauthorized entry to an intranet at a much coarser granularity, an access control policy and mechanisms with associated authentication techniques are used to extend the access authorization at a much finer granularity to individual hosts, databases or application servers within the intranet.

2.2 Security of Transmitted Data

Apart from its role as a protection boundary, a firewall operating at the network layer is capable of providing host–to–host security. The IETF proposal for a security architecture for the Internet [3] at the IP layer (IPSEC) can be used to provide data origin authentication, confidentiality and integrity for connectionless IP datagram delivery. IPSEC defines two security specific headers: IP authentication header (AH) and IP encapsulating security payload (ESP) header. AH is designed to provide only integrity and authentication for IP datagrams [1] while ESP is designed to provide confidentiality along with integrity and authenticity for datagrams [2] transfered through a network connection. ESP can operate either in *tunnel–mode* by encrypting the complete IP datagram and appending a new clear–text header or in *transport–mode* by encrypting only the upper layer PDU contained in the datagram payload.

For secure communication in IPSEC, end–point nodes must establish a security association (SA) [3]. An SA which include security parameters relevant to a particular network connection is uniquely identified by the combination of an IP destination address and a security parameter index (SPI). An SA for IPSEC is generally one–way and contains following information for use by the receiver:

1. Authentication algorithms, their modes and cryptographic keys used for authentication.
2. Encryption algorithms, their modes and cryptographic keys used for encryption. Also, if an initialization vector (IV) is used, its size.
3. Cryptographic key lifetimes or the key update event trigger.
4. The lifetime of the SA and its IP source address.
5. Security label for the protected data. This may conform with a label hierarchy as used in multi–level secure systems.

IPSEC support interoperability by adopting cryptographic algorithms widely used by the Internet community as the default value set for SAs. However, this default cryptographic algorithm set is based on symmetric key cryptosystems (such as MD5 and DES) and does not provide non–repudiation of IP datagram transmission. IPSEC decouples key management mechanism from the task of secure transmission of datagrams. This separation of two streams of activity allow for greater implementation flexibility.

Firewalls need to process IP datagram headers to obtain packet specific information such as source and destination IP addresses. They also need to process datagram payloads to obtain upper layer protocol specific information such as port numbers and protocol types. Such information is used by the firewall to perform appropriate packet filtering and access control. Use of AH in IPSEC in any mode does not affect the operation of firewalls. However, firewalls will not be able to operate correctly if ESP is used with host–oriented keying without firewall access to the corresponding SA [3].

We assume the existence of a supporting infrastructure for successful implementation of a secure extranet. Major elements of such an infrastructure would be digital public key certificate management facilities, trusted third party key servers and meta directory services for obtaining information about participating intranets, hosts, users and applications.

3 Building a High Speed Extranet

To obtain higher speeds of transmission, we need to use high bandwidth network connections when constructing extranets. Two of the main techniques available are

1. Use of high speed dedicated lines for site interconnection.

   Building an extranet using high bandwidth leased lines to create a fully connected mesh over the WAN is an expensive option as well as decreasing the reliability of the network infrastructure. This method would loose the main advantages of using IP datagram routing as the network transmission technology, low cost and fault resilience, in which routing software chooses an appropriate low cost path for hop–by–hop forwarding of IP datagrams while routing around failed or congested links. Therefore, it shall not be further considered herein.

2. Use of high speed routers to connect to a high bandwidth backbone network.

   This method is feasible as major portions of the Internet are supported by high bandwidth optical transmission links provided by major telecommunication carriers. It also has the additional advantage of preserving favorable properties of IP datagram routing, low cost and error resilience. The remaining major requirement for a successful extranet implementation under this option is a high speed routing device.

The addressing model of the Internet and the format of the IP layer PDU results in considerably low performance for the two main tasks carried out by routers:

Packet forwarding: routing table lookup is a sequential search for a longest matching prefix over a relatively large address space.

Packet copying: as IP datagrams are variable size units, copying from an input port to an output port of a router involves computationally intensive buffer management activity.

In comparison, ATM switches perform the corresponding tasks with a much higher throughput:

Cell forwarding: virtual circuit (VC) table lookup is a direct indexed access.

Cell copying: ATM cells are small fixed size units facilitating fast hardware switching between ports and simpler buffer management.

The common approach to building high speed routers has been to integrate link layer (layer 2) switching with network layer (layer 3) routing. The various schemes based on this technique include IP switching [9, 10], Tag switching [14], IP over ATM [12] and CSR [4]. The higher throughput of such a hybrid router is achieved by efficient cut–through switching of fixed–size cells at layer 2 to bypass the slow routing of variable–size packets at layer 3.

The rest of this section is dedicated in the research on IP switching and security. First, an overview is provided of IP switching and then the implications of IP switching to the network layer security are discussed.

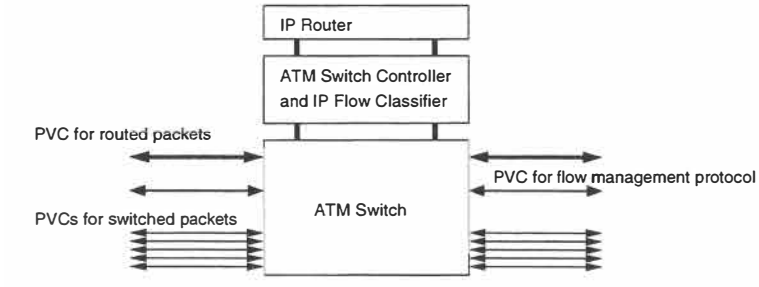


Figure 2: Block structure of the IP switch

3.1 Overview of IP Switching

The operation of an IP switch is based on the concept of a sequence of IP datagrams, termed a *flow*, characterized by a set of common attributes such as source and destination IP addresses, protocol type, port number, etc. The first few datagrams belonging to a flow are routed by the IP switches as in normal datagram routing (*full routing*) at layer 3 and the flow management software in IP switches profiles this flow of datagrams to decide on its suitability for direct switching (*cell streaming*) at layer 2. In Ipsilon IP switches this process is handled by the general switch management protocol (GSMP) [7] and Ipsilon flow management protocol (IFMP) [8]. The decision to build an end–to–end ATM VC for a particular flow is first made by IP switches towards the destination address (downstream) and the connection establishment gradually propagates towards the source address (upstream). Ipsilon's IP switches does not use standard ATM switching protocols and software that conform to ATM–Forum or ITU–T standards and instead use GSMP for switch control and IFMP for link–by–link VC creation and management.

In the cell streaming mode, per–packet processing overhead of full routing is reduced through simple label swapping using connection state tables maintained at each IP switch. This is a compromise between the state–less packet routing in which a routing decision is made for each packet and the state–full (or hard state) packet switching in which an end–to–end connection is pre–established for a complete packet flow. The routed packets are transmitted between IP switches using a default ATM permanent VC (PVC) after encapsulating with a logical link layer/subnetwork attachment point (LLC/SNAP) header for an ATM adaptation layer–type 5 (AAL5) frame. The switched flows are assigned a PVC from a pre–established pool of VCs that each IP switch maintains with its adjacent IFMP–compliant peers (see figure 2). All routed packets are reassembled at the router for routing and flow classification decision making. The IFMP redirect messages sent by a downstream switch to an upstream switch initiates the VC for the switched link for direct hardware switching of the IP datagrams after encapsulating to an AAL5 frame and segmented in to ATM cells.

As shown in figure 3, each local subnet uses an IP switch as both the secure gateway and the border router to connect to the Internet. The maximum performance advantage of IP switching can be obtained only if there is an end–to–end network path through intermediate IP switches. Otherwise, only partial speed increases can be gained for datagram delivery with some segments of a path being switched at layer 2, while remaining segments are routed at layer 3. This scenario is illustrated by the routes indicated in figure 3 showing multiple redundant paths between the subnetworks belonging to the extranet. While each subnet can make an end–to–end connection

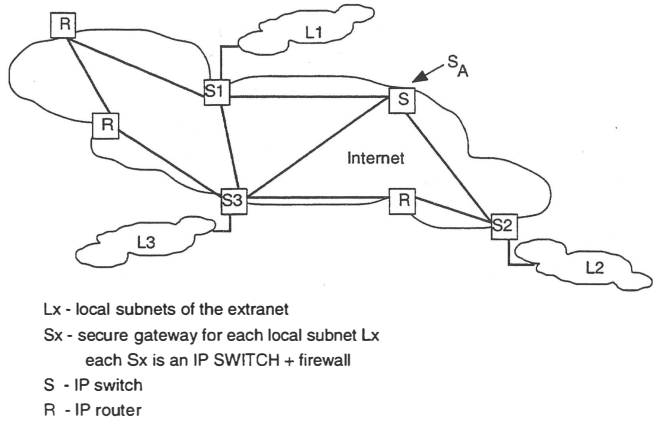


Figure 3: Building high speed extranets using IP switches

through IP switches, if the IP switch S_A were to fail, all datagram traffic to and from subnet L_2 will have to be routed at IP switch S_3. The expected high performance of IP switches can be fully achieved only in a network of IP switches that interconnect with each other to run the IFMP protocol to provide end–to–end layer 2 bypass for packet flows. A simulation study in [6] shows an appreciable performance improvement for a network of adjacent IP switches with a high percentage of datagrams being switched in many of the simulated environments.

3.2 IP Switching and Network Layer Security

If the screening router of a firewall is implemented using an IP switch, it will be able to filter only those packets that are processed under full routing mode. Once the flow of packets is switched over to cell streaming mode, the router no longer has access to the packets for filtering. Therefore, when we bypass (or cut–through) layer 3 processing to achieve higher speed, some of the firewall safeguards will also be bypassed. However, in the extranet model shown in figure 3, IP switches are used as the edge routers of the subnets and receive all outgoing packets and reconstructs all incoming packets. Therefore, the IP switch can perform all firewall related packet filtering based on the IP and TCP header information. Furthermore they can execute IPSEC authentication and encryption tasks on the received packets (after reassembly) and transmitted packets (before segmentation).

When IP datagrams are switched on an ATM VC, the header fields used for flow classification (IP addresses, protocol type, etc) are removed and a compressed header is used for encapsulating the datagram to a AAL5 frame. The removed fields are stored and associated with the corresponding VC for use in header reconstruction at the time reassembling the cells into a packet. This is done to provide an additional measure of security so as to prevent an attacker from establishing a connection to an allowed port through a firewall and then changing the header fields of packets sent through that VC tunnel to gain access to unauthorized hosts or ports [10].

4 An Integrated IP Switch Firewall

In section 2 we have discussed the construction of extranets using standard building blocks (such as Firewalls and IPSEC) available for use in the Internet. In section 3 we discussed improving the network throughput of extranets by replacing standard packet routers with hybrid IP switches. Although switching can increase network bandwidth, continuing use of IPSEC by the IP switch based approach does not address improvements in the efficiency of cryptographic processing needed for secure high speed communication. In this section we propose a modification to the operation of an IP switch to improve its efficiency with respect to security processing.

For secure IP datagram transmission between edge routers functioning as security gateways for extranets, IPSEC must be used in tunnel mode where datagram transmission over the public WAN is always between the gateways through secure virtual tunnels. This allows a gateway to act as a secure proxy for the trusted hosts inside the local subnet and locally deliver or receive datagrams to or from the hosts. Even with secure tunnels, secure datagram transfer and processing is still done on a per–packet basis as the IP model of communication is stateless. However, once a flow is identified for a sequence of datagrams, IP switches begin to maintain soft–state for those switched packet transfers. This gives us an opportunity to increase the speed of security processing by amortizing the cryptographic computational costs over a longer sequence of datagrams rather than on a single datagram as is the case in stateless IPSEC. We achieve this by providing security functionality for the packet stream at the ATM cell level rather than at IP packet level.

It is important to note that the above suggested scheme can be implemented only if the switched flow is established on an end–to–end basis from the source gateway to destination gateway giving a continuous soft–state for the flow. If packet routing is done in intermediate segments of a link, then it is not feasible to map the ATM cell level security association into the IPSEC datagram level security association as intermediate IP switches cannot perform authentication and encryption on behalf of end–point gateways to reconstruct the packets. Above scheme requires several additions to the existing switch and flow management protocols. We briefly outline the main points below

Establishing an ATM cell level security association

1. When a *source gateway* IP switch receives an IFMP redirect message from its downstream neighbor, it will establish the switched flow as per the standard protocol and additionally send a new IFMP *probe request message* to determine if the established flow is end–to–end.

2. If an intermediate IP switch receives a probe request message for a flow that it had not switched onto one of its downstream neighbors, it will send a *negative probe reply message* to the upstream switch. Otherwise, it will forward the probe request onwards to the appropriate IP switch.

3. If an intermediate IP switch receives a probe reply message from a downstream IP switch, it will forward it to the upstream IP switch of the associated flow. If the intermediate IP switch has terminated the upstream switched flow, it will simply discard the received probe message.

4. When a *destination gateway* IP switch receives a probe request for a switched flow, it will send a *positive probe reply message* to the upstream switch.

Using a suitable time–out value and a retry policy, above sequence of steps will enable a source gateway to determine the existence of an end–to–end switched flow. The payload of the probe request message can be used to transmit the set of SA parameters for the associated flow. Conversely, the payload of the probe reply message can provide an acknowledgment for the received SA.

Flow encryption key and secure key transport

Apart from the SA, the payload of the probe request message can securely transport a cryptographic key. This flow encryption key (FEK) is for the encryption of IP datagram payloads to provide confidentiality while being transmitted over the switched link. Also, the FEK in combination with a suitable keyed hash function can also be used to generate a message authentication code (MAC). The MAC can be generated either on a per–datagram or a per–flow basis. For applications such as Telnet in which packets are used interactively, each datagram needs to carry a MAC to ensure its authenticity and integrity. For applications such as FTP, an entire flow can have a single MAC to adequately secure the transmitted content. This MAC generation policy is influenced by application layer considerations and can be parameterized into the SA.

Another important aspect is the secure authenticated transportation of the FEK from the source gateway to the destination gateway. IP switch control messages used by the Ipsilon designed protocols are formatted to fit into single ATM cells for efficient transmission and processing. Secure transmission of a symmetric key of reasonable length (e.g. 70 bits) in a single ATM cell utilizing traditional public key crypto systems such as RSA and ElGamal is not possible due to the very long cryptogram (more than 512 bits) generated by those algorithms in the sign and encryption process. However, the new public key primitive, Signcryption [16], can generate a cryptogram that can fit into the limited payload (maximum of 384 bits) of an ATM cell. Therefore, the flow management protocol can be augmented to efficiently carry a FEK in a single protocol message.

Operation of the extended security protocol

We use IPSEC to provide secure communication when the datagrams are routed or switched by IP switches prior to the establishment of an end–to–end switched flow using probe messages. The security related processing under this mode is external to the IP switch and functions separately in the firewall. However, once an end–to–end SA is established, the security processing needs to be closely integrated with the switch and flow management protocols. For instance, if an intermediate IP switch discontinuous the end–to–end flow by issuing an IFMP reclaim message and reverting to packet routing, then the source gateway must be notified to change over to IPSEC style security processing. For this purpose, in the extended security mode, VC tables maintained by IP switches must be tagged to indicate if they belong to secure flows so that IP switches can send appropriate control protocol message to their upstream neighbors if a VC secure flow is reclaimed or lost due to switch or link failure in a downstream neighbor.

5 Conclusion

Both high speed and security in networking are expensive features to implement and organizations willingness to bear this cost is tied to the new applications enabled by the availability

of secure high speed networks. Corporations that are transforming their business processes to take full advantage of on–line electronic commerce capabilities are the most likely candidates to invest in this technology. It is with this view that we have focused our attention on providing integrated high speed and security for extranets. Security and high speed are properties that are generally orthogonal to each other. This is mainly due to the heavy computational costs associated with the cryptographic primitives use to implement security mechanisms. Therefore, in secure high speed networking we aim to increase network speed while reducing the overall cryptographic computational costs.

The functional integration of firewall and IP switch outlined in section 4 require further investigation to address many operational issues that may arise due to the modification required in existing protocols. A significant disadvantage of designing a security network system optimized for speed by utilizing non–standard techniques is the resultant loss of generality for the entire system. However, this would be a carefully considered implementation decision a system administrator will have to make when building a high speed extranet.

References

[1] R. Atkinson. *IP Authentication Header (AH).* IETF RFC 1826, Aug 1995.

[2] R. Atkinson. *IP Encapsulating Security Payload (ESP).* IETF RFC 1827, Aug 1995.

[3] R. Atkinson. *Security Architecture for the Internet Protocol.* IETF RFC 1825, Aug 1995.

[4] H. Esaki, K. I. Nagami, and M. Ohta. High speed datagram delivery over internet using ATM technology. In *Proceeding of the Networld+Interop*, Las Vegas, NV, Mar 1995.

[5] R. Gareiss. Industrial–strength extranet. *Data Communications Magazine*, pages 71–82, Jun 1997.

[6] S. Lin and N. McKeown. A simulation study of IP switching. In *Proceeding of the ACM Sigcomm*, Cannes, France, Sep 1997.

[7] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. C. Liaw, T. Lyon, and G. Minshall. *Ipsilon's General Switch Management Protocol Specification Version 1.1.* IETF RFC 1987, Aug 1996.

[8] P. Newman, W. L. Edwards, R. Hinden, E. Hoffman, F. C. Liaw, T. Lyon, and G. Minshall. *Ipsilon Flow Management Protocol Specification for IPv4.* IETF RFC 1953, May 1996.

[9] P. Newman, T. Lyon, and G. Minshall. Flow labeled IP: A connectionless approach to ATM. In *Proceedings of the IEEE Infocom*, pages 1251–1260, San Francisco, CA, Mar 1996.

[10] P. Newman, G. Minshall, T. Lyon, and L. Huston. IP switching and gigabit routers. *IEEE Communications Magazine*, pages 64–69, Jan 1997.

[11] R. Oppliger. Internet security: Firewalls and beyond. *Communications of the ACM*, 40(5):92–102, May 1997.

[12] G. Parulkar, D. C. Schmidt, and J. S. Turner. IP/ATM: A strategy for integrating IP with ATM. In *Proceedings of the ACM Sigcomm Symposium on Communications Architectures and Protocols*, Cambridge, MA, Sep 1995.

[13] J. Postel. *Internet Protocol.* IETF RFC 791, Sep 1981.

[14] Y. Rekhter, B. Davie, D. Katz, E. Rosen, and G. Swallow. *Tag Switching Architecture Overview.* IETF RFC 2105, Feb 1997.

[15] L. Zhang, S. E. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A new resource ReSerVation Protocol. *IEEE Network Magazine*, 9(5), 1993.

[16] Y. Zheng. Digital Signcryption or How to Achieve Cost(Signature & Encryption) $\ll$ Cost(Signature) + Cost(Encryption). In *Advances in Cryptology - Crypto'97*, Lecture Notes in Computer Science. Springer–Verlag, 1997.

A Taxonomy and Overview of Information Security Experiments[1]

Erland Jonsson
Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg
SWEDEN
T: +46 31 772 1698, fax:+46 31 772 3663
email: erland.jonsson@ce.chalmers.se

Lech J. Janczewski
School of Business and Economics
The University of Auckland
Private Bag 92 019
Auckland, New Zealand
T: +64 9373 7599, fax:+64 9373 7430
email: l.janczewski@auckland.ac.nz

Abstract

The aim of the research in this paper is to make a survey of the role of experiments and practical project work in the discipline of Information Security education. Thus, we have made a world-wide inquiry to gather information on existing experiments. A few of these are presented in some detail, to give the reader a feeling for what is available. Furthermore, we suggest a taxonomy for such experimentation and classify the existing experiments accordingly.

Keywords: Security, experimentation, education, action learning, classification.

1. This is a revised version of a paper that was presented at IFIP/SEC'97, Copenhagen, 1997-05-14--16.

1. Introduction

Due to the increasing importance of Information Security education, there have been some attempts to review the discipline and to create a common framework for how to handle related issues. Thus, in July 1995 the Erasmus Bureau published a review of university programmes on Information Security [ERASMUS 1995] followed by a proposal for an Information Security curriculum [ERASMUS 1995b]. To our knowledge, this set of publications represents the first systematic attempt to review Information Security education and to suggest a co-ordinated university program. However, these publications do not define the delivery methods, and it seems logical that the research phase to follow should deal with this.

An analysis of the ERASMUS project's publication [ERASMUS 1995] brings us to several quite interesting conclusions. One is the following:

The review of the existing programmes in the field was based mainly on what is offered at one Australian and seven European universities. It shows that these universities are using over 140 different textbooks. Among the publications listed, only one textbook, [Pfleeger 1989], is used at four institutions, one publication, [Muftic 1989], at three locations and 12 publications are used at two universities. The rest of the texts are limited to only one university institution. It is obvious that at the present there is not a great deal of coordination or exchange of information about contents and method of delivery of Information/Data Security subjects. Research on some aspects of delivery methods would be useful.

Therefore, during the 1996 IFIP SEC'96 conference, the IFIP WG 11.8[2] discussed an interesting topic: to what extent is the data security education at university level supported by practical activities, demonstrations, experiments or projects? This discussion, along with the other factors above, became a launching pad for this research, which discusses the possible ways of enhancing Information/Data Security presentation with practical experiments. The present paper covers the rationale behind conducting the experiments, introduces a classification of experiments and lists examples of experiments that might improve the quality of the teaching of the subject.

In the following, section 2 explains the scope of the topic and section 3 gives a framework for the experimental approach. The aim of the research presented is to systematise the role of experiments in teaching data security topics. Such a subject can not be dealt with without presenting experiments already introduced by vario us university organisations. We therefore contacted about 30 universities on five continents and asked about the contents of such experiments. The result is summarised in section 4. A brief evaluation of the data is made in section 5, and section 6 suggests possible directions for future work. Section 7 concludes the paper.

---

2. Working group WG 11.8 operates under the auspices of the Technical Committee TC11 of IFIP and concentrates on issues of data security education.

2. Scope

2.1 The action learning approach

In recent years there has been a growing interest in the *action learning* approach to education. Since 1990, International Congresses on Action Learning, Action Research and Practical Management (Brisbane 1990, Brisbane 1992, Bath 1994, Bogota 1996) have been held on this topic every two years, where scholars from all over the world discuss this approach to education.

According to [Revans1992], [Revans 1984], action learning is a process by which groups of people (whether managers, academics, teachers, students or 'learners' generally) work on real issues or problems, carrying real responsibility in real conditions. The solutions they come up with may require changes to be made in the organisation and they often pose challenges to senior management, but the benefits are great because people actually own their own problems and their own solutions [Zuber-Skerritt 1990].

The action learning approach seems to be ideally suited to studying data security problems. While this discipline does have some highly theoretical parts (such as cryptography), in the majority of cases data security issues are very practical, and are implemented in the real life situation by the developers themselves.

Action learning is based on the Experiential Learning Cycle develop by [Kolb 1994]. See Figure 1.

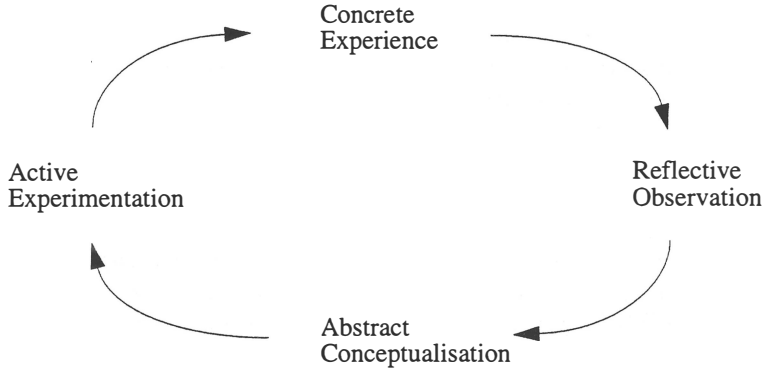


Figure 1: The Experimental Learning Cycle

This cycle indicates the importance of active experimentation, and as a logical extension, significant parts of data security teaching should be based on experimental learning.

2.2 Difficulties with security experimentation

There are several reasons why conducting experiments in the field of data/information security is difficult. The discussion below gives the most important factors supporting that statement:

Information/data security is a rapidly changing discipline. In most cases experiments require lengthy preparation and academics investing great effort in preparing them wish to run them for several years. In the case of data security, this is almost impossible. On the contrary, the development of an experiment increases the effort required to deliver the topic. A good example would be the issue of viruses. Almost all dogmas about them have been changed in recent years. For instance, the arrival of macro viruses invalidated the well known statement that viruses are limited to one platform, i.e., PC viruses do not spread into the Mac world and vice versa. Hence preparation of a demonstration on virus properties requires a careful following of developments in this field and proper updating of the experimental content every year.

Data security experiments generally deal with very sensitive issues. A data security experiment may reveal weak spots in the security armour of an organisation and it might be used against an organisation in many ways: through direct attack or public exposure. Business organisations are well aware of that, and a great deal of persuasion is usually necessary to involve them in a data security experiment. Some time ago, when the issue of data security was relatively new, participants were generally more eager to be involved than they are today. In 1991 the University of Auckland conducted a survey of data security arrangements among local community enterprises. Approximately one hundred organisations were approached and the response rate was 56%. The same research group wished to perform a follow-up of that survey in 1995. Unfortunately the project had to be abandoned, as the response rate was about 5%(!). It was clear that Auckland's business community was alien to this research.

Data security research, by definition, probes the proper functioning of a system. Thus it may happen quite often that an experiment that is "successful" from a researcher's point of view may have quite a disastrous results on an evaluated system. For example, students examining the efficiency of a password system may accidentally gain access to sensitive data or suspend the functioning of the whole system by their actions. Data security experiments should be well protected against these types of calamities.

2.3 Legal issues

Apart from being familiar with technical problems, a data security researcher should be aware of the legal problems resulting from the experiments he/she is conducting. Legal problems focus on the possible violation of privacy laws or similar legislation. An example would be: during a workshop on eavesdropping techniques the students tap highly confidential information. Using that data for any purpose outside the data security research is, of course, forbidden. However, in many countries, permission must in any case be obtained to listen or tap such transmissions.

3. A taxonomy for security experiments

Under the terms of this research, an *experiment* is defined as any activity which is outside the typical lecturing environment, in which a lecturer tells the audience about the theory or practice of the subject. This section suggests a classification of such experiments along three axes: *degree of applicability, degree of innovation* and *level of generalisation.* These are explained in the following.

In terms of ***degree of applicability,*** or distance from reality, the experiments may be of three types, denoted *D, L* and *F,* for *DEMONSTRATION*, *LABORATORY* and *FIELD WORK,* respectively.

D. Conducted by the staff (DEMONSTRATION).

This type of experiment assumes the audience to have a passive role. The lecturer or guest speaker demonstrates the practical side of the addressed course item being addressed. For instance, a lecturer would present (during a lecture) the practical functioning of the reference monitor by connecting to a server and demonstrating various access rights. This type of presentation might take place at any location: it could be arranged during lecturing time, during a session in a university laboratory or at a real business/industrial organisation.

The common denominator is the same: students are passive and demonstrators are active during the conducting of the experiment.

L. Conducted by students in an artificial environment (LABORATORY)

This type of experiment assumes that the students have an active role and are asked to investigate a problem by themselves. The role of the staff is to explain the background, help in case of difficulties etc., but to stay away from the actual experiments. Experiments might be aimed at confirming some theoretical aspects of information security, for instance investigation of the time required for encryption/decryption of a text, or to solve some practical problem, for instance design of an access mechanism.

F. Conducted by students in the real environment (FIELD WORK)

Field Work experiments assume that students are asked to study a problem in a real-life organisation. Such an activity might, for instance, be investigation of the perimeter controls or design of a data security policy for an existing organisation.

The above classification is based on the distance between the participants, i.e., students, and the reality investigated; from a totally passive role (DEMONSTRATION) to dealing with very real problems (FIELD WORK). It implies that DEMO exercises could generally be presented without special preparatory work by students, while FIELD WORK is impossible without that.

The second way to classify experiments is to evaluate the ***degree of innovation*** of the ***object(s)*** of a particular activity. Examples of objects are hardware and software tools, mechanisms, protocols, set of rules etc. The most general classification would include the following classes:

U. Use of the object

These exercises are confined to normal use of the object. The goal might be to learn how it functions and in what situations it could be utilised. An example would be a lecturer that, during a class, is using an authentication and verification procedure to access the system in question.

E. Evaluation of the object

Here the exercises are aimed at presenting the object of the existing information security systems in such a way that its function and use can be evaluated and perhaps rated against other similar objects. A good example of such an activity would be a data security audit.

R. Redesign of an object

The purpose of these experiments is to make a new design of an existing object in order to learn about its basic functionality. The intended result is a better understanding of the security problems related to the design of the object as well as to the integration of it into its intended environment.

N. Design of new or improved objects

These exercises are the most difficult in the classification system. They require in-depth knowledge of the problem area investigated, which could result in new, improved designs of existing objects or recommendations aimed at enhancing the security of the installation. In more advanced cases, this class would also include the development of entirely new (and hopefully more secure) objects.

A step from the U to the N class requires an increased amount of knowledge in and experience of the domain.

The final way of classifying experiments is to define their ***level of generalisation.*** At this stage, we suggest the introduction of three classes:

M. Managerial level

Experiments of this class deal with an overall, organisational level of data security issues. A data security audit of a business unit, as mentioned before, may be example of such an activity. Security policy issues also belong to this level.

S. System level

This is the level of the system administrator as well as the level of abstraction experienced by the normal user of the system. It deals with the direct behaviour of the hardware and software system and with the implementation and use of security mechanisms, logging features etc.

T. Technical level

Experiments in this class deal with the low-level design issues of operating systems and data security mechanisms, such as the internal structure of a reference monitor or an authentication protocol.

Taking all the above into consideration, we suggest that all the experiments in the field of teaching data security should be classified by a three-tuple

{X,Y,Z}, where:

- *X* denotes the ***degree of applicability*** (possible types: *D, L* and *F*)
- *Y* denotes the ***degree of innovation*** required (possible classes: *U, E, R* and *N*)
- *Z* denotes the ***level of generalisation*** of the experiment (possible levels: *M, S* or *T*)

This way of classifying the experiments allows us to generate 36 different classes of experiments. Introduction of a classification of this type is a necessity. Each of the classes requires different preparation and different backgrounds among students and lecturers. For example an experiment of a class *{D,U,S}, "demonstration of the use of an object on a system level"*, does not require students to have a great deal of knowledge, whereas participation in the *{F,N,T}* class, *"design of new technical tools in a real environment"*, demands a thorough knowledge of the discipline in order to yield substantial results.

4. Examples of data security experiments

4.1 Introduction

The aim of this research is to systematise the role of experiments in teaching data security topics. Such a subject cannot be treated without presenting experiments already introduced by various university organisations. To do this we contacted about 30 universities on five continents, North America, Europe, Africa, Asia and Australia/Oceania, and asked about the contents of such experiments at their institutions. Our questionnaire is presented in Appendix 1. In this way we received information of about 15 - 20 different exercises/ projects, which we believe is only a small part of the existing ones. Still, this may serve as a sample that can be used for illustrating the principle and give an idea of the range of experiments.

In this section we present examples of these experiments, apply the suggested classification terminology and make comments about them. All the experiments described below are currently being conducted at various universities.

Each experiment is classified, not only according to the taxonomy suggested in section 3, but also with respect to educational level, duration and the effort required (in man-hours) to perform the experiment:

{X,Y,Z}, educational level, duration, required effort

4.2 Experiment No 1: Eavesdropping techniques

Classification: {D,U,M}, graduate, 2 hours, 2 hours

This experiment aims to demonstrate problems related to eavesdropping techniques: measures and countermeasures.

Students have an opportunity to inspect real "bugs", i.e., "hidden" microphones of various types, and how they may be planted in office and home environments. Live demonstrations of devices that listen to analog and digital cellular phones and to pagers are parts of the demonstration.

On the countermeasure side, a non-linear detector is presented in action. A non-linear detector is a sensor that informs the operator as to whether there is a p-n junction (or semiconductor device) hidden within a radius of about 30 cm around the probe. The operation of a frequency analyser is also demonstrated. This device detects all radio transmission, and hence the presence of any radio-transmitting bugs. No preparation is required of the students prior to the demonstration.

4.3 Experiment No 2: Virus hunt

Classification: {D,U,S}, graduate, 2 hours, 2 hours

In this class, laptops are used to demonstrate typical virus activities (boot sector, stealth, polymorphic, macro etc). An analysis is conducted of the system's resources that demonstrates the existence of a virus. Virus detection and cleaning of software are also demonstrated.

Prior to the demonstration, students must attend a two-hour lecture on viruses and virus software, in which virus mechanics and various types of virus scanners are discussed.

4.4 Experiment No 3: Evaluation of system security by means of synthetic intrusions

Classification: ranging from {L,U,S} to {L,R,T}, undergraduate, about 4 weeks, 40-80 hours

The idea behind this project is to increase students' awareness of security by means of letting them find out for themselves how insecure a system can be, which unfortunately is true for many "normal" systems, i.e., systems in which security is not enhanced and thoroughly managed. The students start the project with no special security knowledge. In many cases they do not even know very much about the object system. Their task is to perform as many intrusions as possible and to report *how* they made them and *how much effort* was required in order to achieve the intrusions.

The results of the experiment are also used for research purposes, in particular to investigate methods for modelling and quantifying the intrusion process. Thus, owing to the requirements of this research, no specific time limit for the duration of the experiment is given, but it is implicit that the expected number of man-hours should normally fall in the range of 40 to 80 hours. The experiment requires careful supervision by a supervisor who must ensure that the experiment is carried out in a realistic way, but without disturbing other users or attempting something that would be illegal or unethical.

The outcome of the project is very dependent on the students involved. An interested and skilful student may very well start to develop new program tools, whereas some students may limit themselves to finding and using existing tools for the attempted intrusions. This is the reason why the classification would include classes such as *{L,U,S}*, *{L,E,S}* and *{L,R,S}* to *{L,U,T}, {L,E,T} and {L,R,T}*.

Each student (or group of students) summarises the results of the work in a *final report,* in which all successful intrusions are listed together with the expended effort. The students may also give their personal comments to the experiment, suggest improvements to the system as a result of their experience etc.

4.5 Experiment No 4: Demonstration of cryptological weaknesses

{L,U,T}, undergraduate, 4 hours, 4 hours

The students are presented with encrypted texts and a list of encryption methods, together with some tools for statistical analysis. Each text is encrypted with a different algorithm, but the students have no prior information about which algorithm is used on a specific text.

Statistical tools and other relevant methods, such as the Berlekamp-Massey algorithm, are used to perform a cryptanalysis of the text. The students are supposed to return the key and plaintext to show that they have been successful in the cryptanalysis.

4.6 Experiment No 5: Implementation of cryptographic algorithm

Classification: {L,R,S}, undergraduate, 4 weeks, 32 hours

The students are given the task of writing a program that implements a known cryptographic algorithm, such as a poly-alphabetic one or columnar transpositions. Furthermore, a brief user's manual is written. The function of the program developed is proven by means of demonstrating its function to the teacher and submitting the user's manual.

4.7 Experiment No 6: Data security audit

Classification: {F,R,M} or {F,E,M}, graduate, 3 months, about 100 hours

In this experiment students are required to perform a security audit of a real business organisation. The work is very closely supervised by the staff. In practice the supervisor is a member of the working team. The experiment is divided into three phases:

I. Preparation

Students undergo intensive training on how to perform a security audit. The exercise is the capstone of their two year study of Information Systems. The training includes familiarisation with the auditing methodology and method of conducting an interview.

The management of the organisation to be audited is contacted and permission is obtained to do the audit. Also, if necessary, security formalities are completed (e.g. issuing of badges, signing of nondisclosure certificates etc)

II. Data collection

Data is collected in three ways: personal interviews, reading related documents and observation. All personal interviews are presented for authorisation after the collection.

III. Data analysis

The data analysis usually contains two types of evaluation: consequences and recommendations. In the first part the team states what could happen if the discovered threat were not eliminated and an attack was launched against the organisation. The second part discusses ways of eliminating the security hole.

The final report has a professional appearance and is signed by the members of the research team, including the supervisor. The report is presented to the company as an official university document. The recipients of the reports normally treat them very seriously and in many cases try to implement the recommendations.

5. A preliminary evaluation of the data

Even though the received material is not large enough to be statistically significant, we have made a brief evaluation of it to see whether there is a tendency towards specific classes of experiments. One of the problems in this work is that some of the larger experiments contain elements of more than one class, e.g., experiment numbers 3 and 6, and may thus be regarded as multi-class experiments. In general, the distribution over the classes will be different if all the classes in a multi-class experiment are considered instead of identifying only one class, e.g., the most common one, in each experiment. However, a good correlation was found between the two ways of calculating, at least with the material available so far. The results below are thus valid for both cases.

In the applicability class, it turns out that laboratory experiments, coded *{L,*,*}*, are by far the most common. Three experiments of four belong to this class. This may not be very surprising, since the laboratory is the traditional place for conducting experiments, although it could have been thought that security experiments might have been an exception to this.

As regards the innovation class, there is a tendency towards an even spread between "use of" or "redesign of" the object, i.e., *{*,U,*}* or *{*,R,*}*, whereas "evaluation of" the object, *{*,E,*}*, is less common and "design of new improved objects", *{*,N,*}*, is quite infrequent.

The most common level of generalization for the experiments is the system level, *{*,*,S}*, which is as common as the two other groups, *{*,*,M}* and *{*,*,T}*, together. Obviously, it is easier or more natural to develop experiments which are on the system level, and thus more or less directly referring to the user, than to go up or down in the hierarchy. The management level would require an overview and the technical level a knowledge of details not shared by all students. A plausible interpretation of this fact is that business and management educations have a focus on the management level and vice versa.

In summary, the most "typical" experiment is an *{L,U,S}* or *{L,R,S}*, and these two classes together amounted to almost half of the total number.

6. Discussion and future work

While the present analysis covers most of the classes introduced in the taxonomy - from demonstrations conducted during regular lectures to substantial field work involving close co-operation with industry or other external organisations - it is not surprising that a vast majority of the experiments were performed in laboratories and aimed at redesigning or using well-known objects. Here, an interesting question is whether this outcome reflects an optimal set-up of experiments or is the result of some other condition, e.g., that the experiments carried out are simply those that were easiest to organise. We suggest that future investigation attempts to clarify this issue. Another related issue would be to establish the results of the experiment, preferably in quantitative terms, such as student satisfaction or learning effect. It is clear that there are a number of factors that might influence the result and that must be considered.

Examples are:

- attitude of students and staff towards conducting the experiments.
- quality of the experimental leader (e.g., staff, students, expert, tutor etc.)
- level of studies
- the course material used

Finally, we would like to point out that the descriptions we received of experiments carried out at various universities should preferably be made available to all information security educators. We suggest establishing a databank for the collection of descriptions of such experiments.

7. Conclusions

This paper is a first attempt to present a rationale behind enhancing information security studies with experimentation. A classification method was developed and typical experiments presented and classified. We can also conclude, on the basis of the full material received, that this type of experimentation is very much in line with present trends in engineering education. Not only is it a good example of action learning, but it also incorporates substantial elements of innovative teaching and interdisciplinary approaches, as discussed in [Smith 1991] and [Yngström 1996].

The results so far are quite rewarding. Still, they call for further research to be undertaken, both to gain a better understanding of the experimental process as such and to put it into an educational context.

Acknowledgement

We would like to thank all of our contributors without whose help this research would not have been possible. The page limit of the paper prevented us from incorporating all experiments.

References

[ERASMUS 1995] Gritzalis, D. (Ed), *University Programmes on Information Security, Dependability and Safety*, European Commission, Erasmus ICP, Projekt ICP-94(&95)-G-4016/11, Report IS-CD-3c, Athens, July. 1995.

[ERASMUS 1995b] Katsikas, S., Gritzalis, D. (Eds), *A Proposal for a Postgraduate Programme on Information Security, Dependability and Safety (Syllabus)*, Version 2.2, European Commission, Erasmus ICP-94(&95)-G-4016/11, Report IS-CD-4a, Athens, Sept. 1995.

[Kolb 1994] Kolb, D. Experiential Learning, Experience as the Source of Learning and Development, Prentice-Hall (1984).

[Muftic 1989] Muftic, S.: Security Mechanisms for Computer Networks, Ellis Horwood Ltd, England, ISBN 0-7458-0613-9, 1989.

[Pfleeger 1989] Pfleeger, C. P.: Security In Computing, Prentice Hall International, Inc. ISBN 0-13-799016-2, 1989.

[Revans 1984] Revans, R., The Sequence of Managerial Achievement, MCB University Press, Bradford (1984).

[Revans1992] Revans, R., The Origins and Growth of Action Learning, Chartwell-Bratt Lty, Bromley (1982).

[Smith 1991] Smith, R. A. (Ed.), "Innovative Teaching in Engineering", Ellis-Horwood (1991). ISBN 0-13-457607-1. pp. 3-40, 253-294.

[Yngström 1996] Yngström, L., IT Security and Privacy Education. In Proc. of the 12th International Information Security Conference, IFIP/SEC'96, Samos, May 21-24, "Information Systems Security: Facing the information society of the 21st century". Chapman&Hall. ISBN 0-412-78120-4. pp. 351-364.

[Zuber-Skerritt 1990] Zuber-Skerritt, O., Action Research For Change and Development, Centre for the Advancement of Learning and Teaching, Griffith University, Brisbane (1990).

Appendix: Questionnaire.

To: Teachers of Computer Security and other interested parties,
RE: Request for data on practical security experiments

[General information on the research project - not included]

****

DATA SECURITY EXPERIMENT

University.....
Faculty.....
Department.....
Course name.....
Course level (undergraduate, graduate, etc).....

Experiment type (please circle) DEMO LAB FIELD

Experiment duration (in min, hours, days, etc).....

Experiment goal.....
.....

Experiment description.....
.....

Assessment method (if appropriate).....
.....

****

A Preliminary Evaluation of the Security of a Non-Distributed Version of Windows NT

Hans Hedbom, Stefan Lindskog
{Hans.Hedbom, Stefan.Lindskog}@hks.se

Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, SWEDEN
and
Department of Computer Science
University of Karlstad
S-651 88 Karlstad, SWEDEN

Erland Jonsson
erland.jonsson@ce.chalmers.se

Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, SWEDEN

Abstract

In this paper we present a preliminary evaluation of the security of a non-distributed version of Windows NT. The objectives of the work are twofold: first, to learn more about the security system; and, second, to find out how secure the system actually is. Thus the architecture and security mechanisms of Windows NT have been studied. Furthermore, the paper contains a few examples of successful intrusions on the target system, which was a standard personal computer with Windows NT Workstation 3.51 and one with NT Workstation 4.0, both working in a stand-alone mode. We have also found some evidence that other, more severe security flaws exist in the system.

1.0 Introduction

Computer security is traditionally defined by three attributes: (1) confidentiality (or secrecy), (2) integrity, and (3) availability [11] also known as "the CIA". **Confidentiality** is the aspect of keeping information protected from unauthorized users. **Integrity** implies that data only can be modified by authorized parties. Finally, **availability** means that the services are provided to any authorized user. A violation to any of these attributes is considered to be a successful attack.

Windows NT was designed with security in mind. In fact, it has been classified as a C2 level operating system (OS) by the National Security Agency (NSA) [8]. There were two major goals with this study: first, to find as many vulnerabilities as possible in NT, within a limited time. Second, to gather information about NT and its flaws.

A common method to evaluate the security of a system is to use a so called *Tiger Team*. For example, see [1], which describes a Tiger Team analysis of VM/370 OS from the mid-seventies. Such a team is very skilled and has deep knowledge about the system and its potential vulnerabilities. In our case, however, there is a difference. We were novice NT users.

There is not much in-depth literature on NT or its security design. However, there is one evaluation performed by the NSA, and the corresponding report has been issued by the National Computer Security Center (NCSC) [8]. The evaluation is mostly based on the design of the system as defined in the Orange Book [13]. We have, on the other hand, studied the operational security of NT using penetration experiments. These may reveal vulnerabilities in the design, implementation as well as in the installation.

Other penetration experiments have been carried out at the department of Computer Engineering at Chalmers University of Technology, the most similar being a security analysis of a secure database [6]. However, most of the previous studies differ from the present. Firstly, the object systems were different: a networked UNIX operating system [9], [10] and a PC Network [4]. Secondly, the attackers were final year university students. Thirdly, the attackers had to follow a number of quite specific rules, since the result of these studies were used for mathematical modeling purposes.

Section 2 provides an overview of NT. Section 3 describes the security features, whereas section 4 focus on utility programs for NT. Section 5 presents some intrusion experiments and section 6 concludes and gives implications for further work. Last is a list of abbreviations used in this report.

2.0 System Overview

The Windows NT operating system (NTOS) was developed by Microsoft Inc. and was first released in 1992. It is expected to replace Windows 3.x. Unlike Windows 3.x, NT is a full-fledged 32-bit OS with support for: processes, multiple threads, symmetric multiprocessing, and distributed computing. NT uses an object model to manage its resources. Therefore, the term *object* is used instead of resource. Moreover, it is designed to be a secure OS, e.g. one of the goals is to meet the C2 evaluation criteria. A C2 system must implement discretionary access control at user level, and it must provide mechanisms for tracking all accesses (or attempted access) to an individual object [11]. Another C2 level requirement is elimination of residue exposure.

2.1 System Architecture

An OS can be designed in different ways, and the three most common models are [2]:

- the monolithic model
- the layered model
- the client/server model

The structure of NT is a hybrid between the layered model and the client/server model. NT uses the later to provide the user with multiple OS environments (Windows, MS-DOS, OS/2 and POSIX (Portable Operating System Interface based on uniX), see Figure 1.

The executive is the only part of the system that executes in kernel mode, and is divided
into three levels. The lowest level is called Hardware Abstraction Layer (HAL), which provides an abstract view of the underlying machine architecture. The motive for having this layer is to make the system (more) portable.

Above HAL is the microkernel. This is responsible for low-level support for execution, interrupts and exception handling, and synchronization [8].

The top-most layer in the executive consists of a number of components (modules) implementing basic OS services, such as: virtual memory management, object management, process and thread management, I/O management, Interprocess Communication (IPC), and security reference monitoring. Communication between these components works through a set of well defined functions in each component.

2.2 Protected Subsystems

A protected subsystem provides an Application Programming Interface (API) which programs can call [3]. Such protected subsystems are sometimes called servers, or protected servers, and are executed in user mode as processes with certain privileges [8]. When an application calls an API routine, a message is routed to the server implementing the API routine via the Local Procedure Call (LPC) facility. Later, the server replies by sending a message back to the caller. Trusted Computer Base (TCB) servers [8] are protected servers which execute as a process with a SYSTEM security context, which implies that the process possesses a specific token (see 3.0).

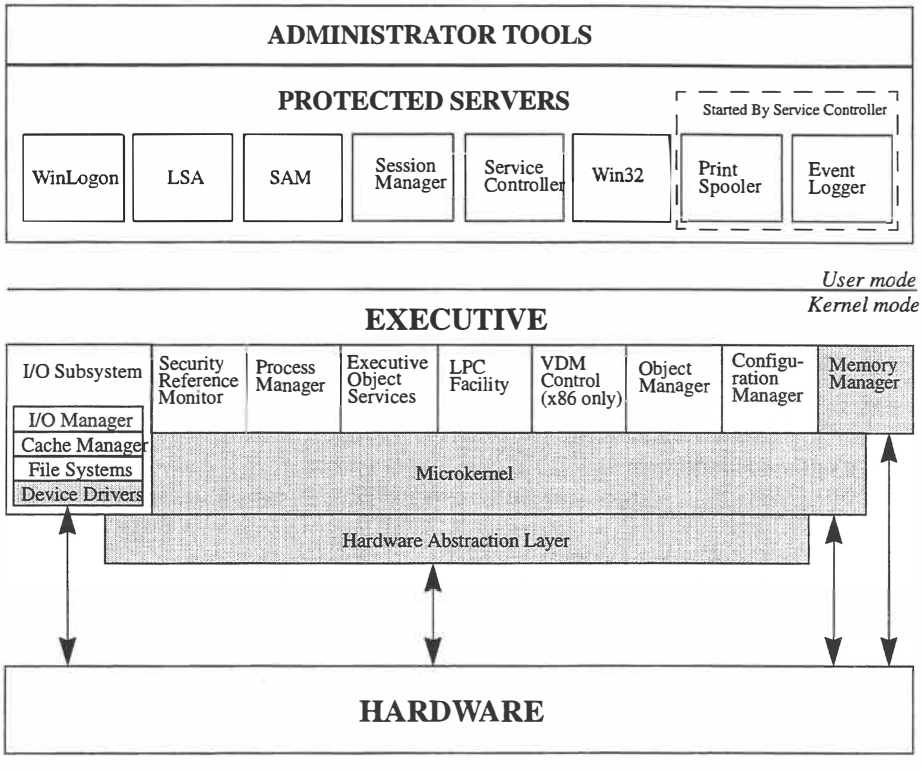


FIGURE 1. Windows NT System Overview [8]

2.3 Users and groups

Under NT, a person that needs access to objects on a computer must have a valid user account. In addition, to access a specific object the user has to have access rights on it. Every user account contains information such as: username, password, full name, logon hours, logon workstations, expiration date, home directory, logon script, profile and account type.

By default there are two accounts: Administrator and Guest. Within the Administrator account, new accounts can be created. To add new user accounts, the User Manager tool is used. This is a standard tool, distributed and installed together with the OS.

Windows NT supports the concept of groups, similar to those of UNIX. Some people [7] is of the opinion that NT groups are more powerful than groups in UNIX. With groups, permissions can be granted to a set of related users, which makes the procedure of granting rights and permissions easier. Moreover, a single user can belong to more than one group at the same time. Furthermore, NT provides a number of built-in groups: Administrator, Backup Operators, Printer Operators, Power Users, Users, and Guest. A user belonging to the group Backup Operators, for example, has the rights to

backup the system. In UNIX by default, it is not possible to do this task without being superuser (*root*). Altogether, this implicates that different types of system administrators, with a different set of access rights can be defined in NT.

In addition to these built-in groups, a system administrator is able to define new types of groups, which can be done in the User Manager.

2.4 File Systems

NT supports multiple file systems, including the FAT (File Allocation Table) file system, the HPFS (High Performance File System), and the NT file system (NTFS) [2]. The FAT file system was designed in 1976 by Bill Gates. Later, it was chosen for the MS-DOS operating system. HPFS is designed by Microsoft for OS/2 1.2, mostly to support the LAN Manager file server. Finally, NTFS is a completely new file system for NT. NTFS is designed to be all things to all people and to include all features of every other file system in common use. In NT, file systems are accessed through file system drivers, which can be dynamically loaded into (and out of) the system. For each supported file system (FAT, HPFS, and NTFS) there is a driver, see Figure 2.

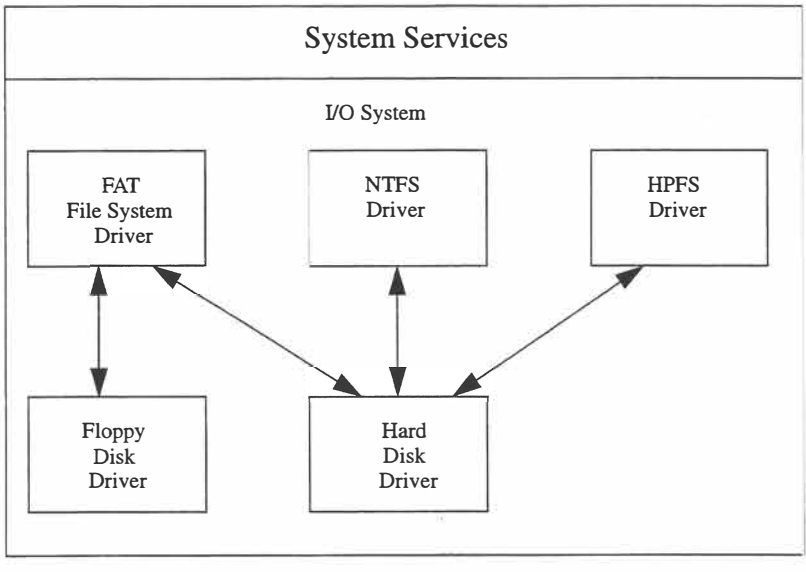


FIGURE 2. Drivers for different file systems

3.0 Security Features

NT provides a unified access control facility which applies to processes as well as other objects in the system. The access control is built around two components, an access-token associated with every process, and a security descriptor connected to every object where interprocess access is possible. When a user logs onto the system she has to provide a username and a password to authenticate herself to NT. If she is accepted, a process and an access token are created for her. The access token is associated with the process. An important attribute in the access token is the Security ID (SID). Which is an identifier that identifies the user to the system in areas concerning security. A process inherits the access token of its creator. There are two reasons for maintaining access tokens [12]:

1. It keeps all the necessary security information in one place in order to speed up access validation.
2. It allows every process to modify its own security attributes, in a restricted manner, without effecting other processes in the system.

It might seem strange at first glance to let the process change its own security attributes, but normally a newly created process is started with all attributes disabled. Later when a thread wants to perform an allowed privileged operation the thread enables the necessary attributes before the operation is executed.

3.1 Subjects and Objects

A subject in the NT system is a process with one or more threads that execute on behalf of a user and objects are all other NT resources that can be accessed and used by a subject. Note that incidentally a process in NT is also called an object. The privileges, user SID and group SID of a user are stored in an access token held by the processes. This token, called the primary token, is used to grant threads different types of access to different objects. Threads can also hold one additional access token called an impersonation token. This is a token given to the thread by another subject which allows the thread to act on that subjects behalf and is a full or restricted variant of that subject's primary token. The information in the token is compared to the information stored in the object's Access Control List (ACL). This comparison is handled by the Security Reference Monitor (SRM) and performed when the object is first opened by the process. The privileges defined and those entities granted these privileges are stored in the Local Security Policy Database managed by the Local Security Authority (LSA) which is also responsible for creating the primary token for a user during logon. The primary token (or restricted variants of it) will later be inherited by the processes created or used by the user. Since the first token is created during login, it is not possible to add or remove privileges while this user is logged on (or rather it is possible to change them but the change will not take effect until next logon). However, they can be enabled or disabled.

3.2 User Login and Authentication

The login procedure in NT is fairly complicated and involves a number of executives and protected servers to authenticate and grant privileges to a user. The details of the login procedure are given in [8], and the following is a shortened version of the events

that take place. The coordinator of the process is the WinLogon server, see Figure 1. When the system is in its initialization phase, WinLogon registers itself at the Win32 server as the login process (see Figure 3), and ensures that no other process has access to the WindowStation object created by Win32. This is done by assigning a security descriptor to the WindowStation with only one entry - the WinLogon SID. It establishes a link to LSA, where it also registers as the logon process, makes the WinLogon Desktop active, and waits.

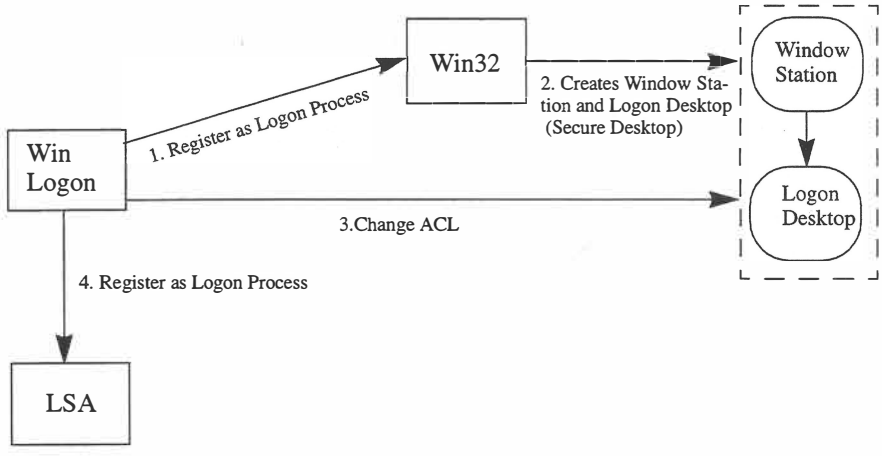


FIGURE 3. Schematic view of the login procedure: step 1 (Numbers state chain of events)

When a user presses the Ctrl-Alt-Delete key combination (called the Secure Attention Sequence (SAS)) WinLogon initiates the identification and authentication process and brings up a secure Desktop that prompts for username and password (see Figure 4). The password is collected by Win32 which encrypts the password with an internal encryption, later decrypts it and passes the password to WinLogon (see Figure 5). When WinLogon receives the username and the password it collects the domain of the user, creates a unique logon SID, creates and suspends a new process, and calls LSA (see Figure 6).

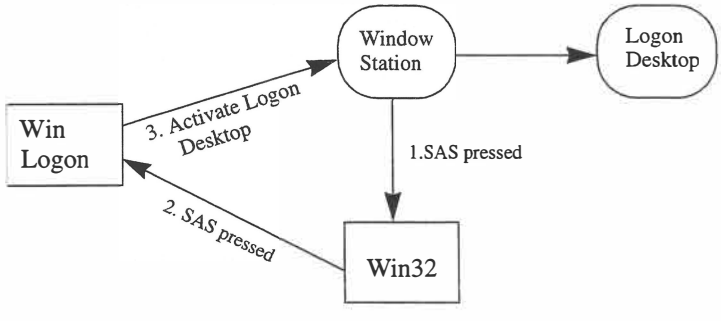


FIGURE 4. Schematic view of the login procedure: step 2 (Numbers state chain of events)

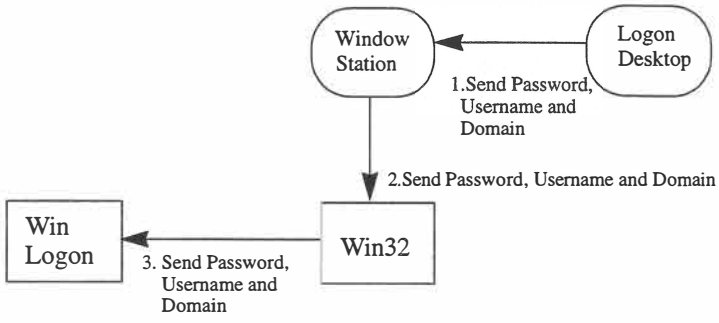


FIGURE 5. Schematic view of the login procedure: step 3 (Numbers state chain of events)

The parameters passed to LSA in the call are those collected and created above and the Authentication package that WinLogon wants to use. Since it is possible to use more than one Authentication package this information is important. Before the password is passed to LSA it is once again encrypted with an encryption known only to WinLogon and the Authentication package.

Once LSA has been called it calls the Authentication package. This is done because LSA can not use some of the parameters because they are encrypted. It is instead the Authentication package that verifies the username and the password. This is done by retrieving some information from the Security Accounts Manager (SAM) database, including the hashed password, the user SID and the group SIDs that are associated with this user (see Figure 6).

The authentication is done by comparing the two hashed passwords with each other. The hashed cleartext password, the username and the hashed case-sensitive password is then stored in a variable supplied by LSA.

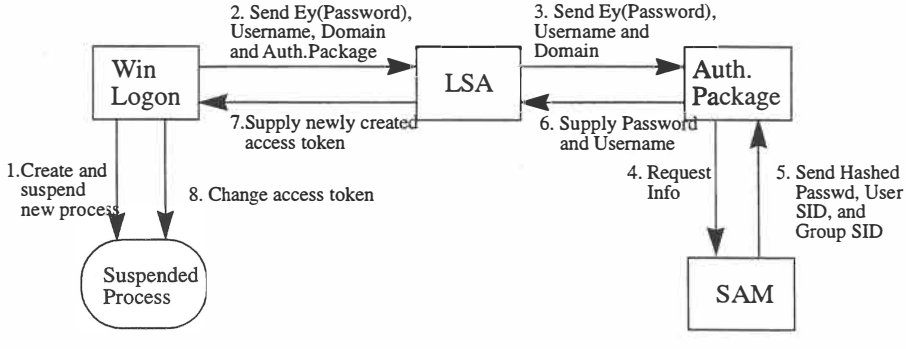


FIGURE 6. Schematic view of the login procedure: step 4 (Numbers state chain of events)

If the authentication package part of the login process succeeds the LSA will check the privileges held by the user and the groups of which she is a member and determine if the user is allowed to perform this type of login (there are three types available: inter-

active, network and service logins). If she has the right privileges the logon SID mentioned above is added to the list of local groups and a default ACL is constructed for the user together with the user's primary token (see 3.1). A handle to the primary token is passed to WinLogon which uses the handle to create an ACL for the user in the Discretionary ACL (DACL) of the WindowStation and to set a new primary token on the suspended process (see Figure 6). After that, the WinLogon creates an application Desktop and a screensaver Desktop and change the DACL on both of them so that WinLogon and the user have access to the two desktops.

If the authentication should fail in any way the LSA will notify the WinLogon Server and the server will display an error message through the Desktop and prompt for new information.

3.3 Access Control Lists (ACLs)

As mentioned before, every named object in NT, together with unnamed processes, threads or token objects have a security descriptor in its object header. The security descriptor has information about the logging for the object and its access rights. The access control is implemented by an ACL. The ACL consists of a list of Access Control Entries (ACEs). An ACE contains a SID and the operation that the owner of this SID is allowed (or disallowed) to perform on the object. ACEs that disallow operations is generally placed before ACEs that allow operations in the ACL.

3.4 Auditing

The auditing in NT is handled by SRM and LSA together with the Event Logger. Different types of events are grouped into event categories and auditing is then done based on these categories. There are seven types of event groups. These are: System, Logon/Logoff, Object Access, Privilege Use, Detailed Tracking, Policy Change and Account Management. For details on event groups see [8]. If auditing applies and what is to be audited is determined by the *Audit Policy* which is handled by the LSA and given to the SRM by LSA.

The auditing is based on audit records constructed on request from the responsible subsystem or server by SRM (in some cases by LSA). Requests from the executive is always carried out while servers need the Audit privilege for SRM to honour there request. The request must be sent for each occurrence of an event. The audit record is then sent to the LSA, which in turn sends it to the Event Logger after it expanded some fields and compressed others. The final writing to disk is made by the Event Logger.

4.0 Utility Programs

In this section, we present some utility programs that have been useful to us in our study of NT. For example NTFilemon was an excellent program that gave us the opportunity to learn more about the activity in the file system, when the system was up and running. We also believe that some of these tools can be used (or misused) by the bad guys in a security attack. All utilities presented here are publicly available on the Internet.

4.1 Windows NT Password Dump Utility (PWDump)

The PWDump utility dumps the password database of an NT computer in the following format:

<user>:<id>:<lanman pw>:<NT pw>:<comment>:<home dir>:

Where <user> is the user name on NT, <id> is the last 32 bits of the SID, <lanman pw> is the Lan Manager password hash, <NT pw> is the NT password hash. These fields are the important ones. The <comment> and <home dir> fields contain information such as the user's full name, description and home directory as specified in the NT User Manager. PWDump can be used on both the local machine and remote machines. However, in order to run the program the user has to log on as Administrator.

4.2 Access to functions and variables in NTOSKRNL (NTExport)

NTExport is a tool that can be used to generate a kernel-mode device driver library, including undocumented functions and variables which are exported from the NTOSKRNL. It supports only NT 4.0 x86 versions of NT. The question is: how can this program find information about these functions and variables? NTExport combines information in the NTOSKRNL.EXE and its corresponding debug symbol file (NTOSKRNL.DBG). The latter can be found under \support\debug\i386\symbols\exe on the distribution CD.

4.3 NT Registry Monitor (NTRegmon)

NTRegmon is a program that displays all registry activity taking place on a NT system. Version 2 of this program can be used on both NT 3.51 and NT 4.0. NTRegmon consists of two parts: a device driver which uses a technique called kernel-mode system call hooking, and a graphical user interface (GUI). When the GUI program is started, it dynamically loads the driver. This registry monitor sees all user level registry activity, and even most kernel registry calls.

4.4 NT File System Monitor (NTFilemon)

NTFilemon can be used to monitor file system activity for NT 3.51 and NT 4.0. This utility consists of a device driver and GUI, like NTRegmon. The device driver is a type of driver known as a filter driver, which means that it layers itself above the file system drivers. After installation, NTFilemon can see I/O requests pass to, and from, the file systems. All types of file system drivers that have an associated driver letter may be

monitored, e.g. FAT and NTFS partitions. The GUI application will automatically load the driver.

4.5 NT Process Monitor (NTPmon)

With NTPmon, information about all process activity on an NT 4.0 system can be gathered and displayed in a window. The utility program consists of a device driver and a GUI, like NTRegmon and NTFilemon. The driver uses a number of undocumented hooking functions, which collects information about activity such as process creation, process deletion, thread creation, thread deletion, and optional context switches. The latter is only present in multiprocessor builds of NT, and is by default disabled.

4.6 Object Manager Name Space Viewer (WinObj)

WinObj is an NT application that displays information on the NT object manager's name space. This means that the program shows information about various operating system components. WinObj uses the native NT API, which, for example, provides routines to allow user programs to browse the name space. It is also possible to query the status of objects located there.

4.7 Keyboard Filtering (ctrl2cap)

The main purpose of Ctrl2Cap is to convert the control (ctrl) key to the shift key, so that the program in itself is not harmful, but the implications inherent in this are. We believe that by extending this driver or writing a similar program it is possible to catch and manipulate any key pressed on the keyboard before NT sees it, which means that it is feasible to snoop, and save passwords when they are entered. It is also possible to filter out the ctrl-alt-delete key combination which would totally lock the workstation since no user will be able to logon to that station. This in turn would cause trouble on workstations shared among several users, or could be used by a malicious user to gain access to the administrator password on his personal workstation.

5.0 Examples of Intrusions

We have found several sites on the Internet that present programs that utilizes flaws that endangers the security of the NTOS. In this section we will give a few examples of some of the these programs and how they could be used to gain unauthorized access or information from a stand-alone NT system.

5.1 Experimental System

The target systems used to test software have been two standard PCs. One with off-the-shelf NT 3.51 Workstation software, and the other with off-the-shelf NT 4.0 Workstation software. The 3.51 version ran on a computer with an Intel 486 processor, and the 4.0 version on a Pentium processor. Both have been operating as stand alone systems so no network support has been used on either of them (see section 6.1). The user profiles used have all been copies of the standard guest account except when special privileges have been needed. On those occasions we have also tested them as Administrator.

5.2 NTCrash

Description. NTCrash is a program written by Mark Russinovich and Bryce Cogswell that exploits certain implementation flaws in NTOSKRNL. It is loaded from NTOSKRNL.EXE and contains the majority of the OS components that are executed in kernel mode. NT programs use the NTOSKRNL by invoking functions through calls to certain libraries (Dynamic Linked Libraries (DLLs)). In some of these calls the parameters are not checked properly. The missing checks are primarily range checks and legality of addresses. By invoking these functions with illegal or *out of range* or *out of bounds* parameters, NT will crash.

Intent. We will try to bring our target system to a grinding halt by invoking this program. What we want to achieve is to force a restart of the system and thereby deny service to others. In the target system, this would only stop our selves from using the system, but if it was executed on a server or a domain server this program could cause denial of service for a number of users as well as potential loss of data.

Result. We executed the program as follows:

```
ntcrash -n
```

After a few seconds the computer crashed, and therefore in accordance with our intentions the attack was a success.

5.3 NTFSDOS

Description. NTFSDOS is a program that makes it possible to read NTFS files under MS-DOS or Windows 95. This means that any file security attributes are bypassed. In the current version, it is only possible to read NTFS files. Indeed, it is a question of time before a program turns up that is also able to modify NTFS files under MS-DOS. The implications of this is that any system with a floppy drive, or which is bootable with another OS, is unsafe.

Intent. We intend to carry out the attack as follows.

1. We will create a MS-DOS boot floppy disk and put a copy of NTFSDOS on it.
2. We will then boot the target system from this floppy.
3. When the OS (now MS-DOS) is up and running, we will start NTFSDOS and hopefully every file on any drive on the target system will be readable by us.

Result. We did as specified above. After that, all files in the NTFS partition were readable.

Comment. If it is possible to read the NT file system, then it is also possible to read the SAM database. Information in this database includes hashed and encrypted passwords, which can be gathered by an attacker once she knows the internal file structure. Both the encryption methods and the way encryption keys are generated are known.

5.4 L0phtcrack

Description. The L0phtcrack (pronounced "loftcrack") is a quite new password cracking program, which can be used to recover both the Lan Manager password and the NT password, stored in the registry. Revision 1 of this program takes as input a file with user information, including both the username and the password. Such a file can be created by the PWDump utility, see 4.1. L0phtcrack can optionally take a dictionary file as input. This type of attack is often referred to as a dictionary attack. Alternatively, L0phtcrack gives the attacker the capability to apply a brute force attack on the entire key space. The utility is distributed with both a graphic and a character user interface.

Intent. We intend to try finding the password of all user accounts on the target system by using L0phtcrack. If the attack succeeds we will have access to resources in the system.

Result. This attempt was not successful, due to the fact that the required PWDump utility has to be run as Administrator. Still, we tried L0phtcrack with the sample password file, see 4.1, and the dictionary file distributed with L0phtcrack. The result of this attack is given below:

User [Guest] account is disabled
User: [BillG] Lanman PW: [YOKOHAMA] NT dialect PW: [YokoHama]
User: [fredc] Lanman PW: [CRACKPOT] NT dialect PW: [crackpot]
User: [william] Lanman PW: [IMPUNITY] NT dialect PW: [impunity]
User: [Administrator] Lanman PW: [SCLEROSIS] NT dialect PW: [ScleROSIS]

The result was given within a few seconds. After that, we made a brute force attack on the same password file on a PC with an 166 MHz Pentium processor running NT 4.0. After 115 hours, the following result was presented:

This build using characters [ABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789 '~!@#$%^&()_-{}"'.] for keyspace. beginning bruter (this may take a while)

User: [Joe] Lanman PW: [ACOLL1171] NT dialect PW: [acoll1171]

Joe was the only user account not cracked by the dictionary attack.

In [5], the authors promise that revision 2 of L0phtcrack will not require the user to be an administrator or have an administrator's password. Moreover, L0phtcrack will be able to access the password database on remote sites.

Comment. The Lan Manager password is padded with NULLs if it is not 14 characters long. This together with the encryption method used creates recognizable patterns if the password contains less than eight characters.

6.0 Conclusions

It is relatively hard to find relevant information about the design and implementation of the NTOS. However, during the course of the study more and more material has become available, so we are confident that this situation will improve. Still, we have found - on Internet and in other sources - various kinds of information, ranging from hints of weaknesses to fully executable utility programs. This has allowed us to perform successful intrusions of various types.

Thus, we have shown that it is indeed possible to violate the security of the Windows NTOS working in a stand-alone mode, even for a novice user. Furthermore, there are implications that there could be more severe security problems within NT. For example, many application programs used today are written for Win32s (the 32-bit extension for Windows 3.x) or Windows 95. Few application programs are designed directly for NT. This implies that they are not supported by the security features available in NT, which obviously opens up for further weaknesses.

In conclusion, the outcome of this study shows that further, in-depth studies are necessary, in order to get a more complete picture of the security of Windows NT.

Acknowledgment

The work described in this paper was supported by Telia Research, Haninge, Sweden.

References

[1] C. R. Attanasio and P. W. Markstein and R. J. Phillips, Penetrating an Operating System: a study of VM/370 Integrity. IBM Systems Journal, pp. 102-116, Volume 16, Number 1, 1976.

[2] Helen Custer, A Grand Tour of Windows NT: Portable 32-bit Multiprocessing Comes to Windows. Microsoft Systems Journal, Volume 7, Number 4, pp: 17-31, Jul-Aug, 1992.

[3] Helen Custer, Inside Windows NT. Microsoft Press, 1993.

[4] Ulf Gustafsson and Erland Jonsson and Tomas Olovssson, Security Evaluation of a PC Network Based on Intrusion Experiments. Proceedings of the 14th International Congress on Computer and Communications Security, SECURICOM '96, Paris, France, pp. 187-203, June 4-6, 1996.

[5] Larry Lange, Hackers keep the heat on Windows NT Security. DVD Technology Seminar, June 10, 1997.

[6] Ulf Lindqvist, T. Olovsson, E. Jonsson, An analysis of a secure system based on trusted components." In Proceedings of the Eleventh Annual Conference on Computer Assurance (COMPASS '96), pp. 213–223, Gaithersburg, Maryland, USA, June 17–21, 1996. IEEE.

[7] Microsoft, Microsoft Windows NT from a Unix Point of View: A White Paper from the Business Systems Technology Series. 1995. Dennis Martin, Windows NT File System Security: An Overview and Comparison, Revision 1.1. November 1994.

[8] NCSC, FINAL EVALUATION REPORT Microsoft Inc.: Windows NT Workstation and Server Version 3.5 with U.S. Service Pack 3. National Computer Security Center, 1996.

[9] Tomas Olovssson and Erland Jonsson and Sarah Brocklehurst and Bev Littlewood, Data Collection for Security Fault Forcasting: Pilot Experiment. Technical Report No 167, Department of Computer Engineering, Chalmers University of Technology, 1993 and ESPRIT/BRA Project No 6362 (PDCS2) First Year Report, Toulouse, France, pp. 515-540, Sept. 1993.

[10] Tomas Olovssson and Erland Jonsson and Sarah Brocklehurst and Bev Littlewood, Towards Operational Measures of Computer Security: Experimentation and Modelling, in B. Randell et al. (editors): Predictably Dependable Computing Systems, Springer Verlag, ISBN 3-540-59334-9, 1995. pp. 555-572.

[11] Charles P. Pfleeger, Security in Computing. Prentice-Hall International, 1989.

[12] William Stallings, Operating Systems, 2nd Edition. Prentice Hall, 1995.

[13] Trusted Computer System Evaluation Criteria ("orange book"). National Computer Security Center, Department of Defense, No DOD 5200.28.STD, 1985.

A Terminology

ACE	Access Control Entry
ACL	Access Control List
API	Application Programming Interface
CIA	Confidentiality, Integrity and Availability
DACL	Discretionary ACL
DLL	Dynamic Linked Library
FAT	File Allocation Table
GUI	Graphic User Interface
HAL	Hardware Abstraction Layer
HPFS	High Performance File System
IPC	InterProcess Communication
LPC	Local Procedure Call
LSA	Local Security Authority
NCSC	National Computer Security Center
NSA	National Security Agency
NT	Windows New Technology
NTFS	NT File System
NTOS	NT Operating System
NTOSKRNL	NT Operating System KeRNeL
OS	Operating System
POSIX	Portable Operating System Interface based on uniX
SAM	Security Accounts Manager
SAS	Security Attention Sequence
SID	Security ID
SRM	Security Reference Monitor
TCB	Trusted Computer Base
VDM	Virtual Dos Machine
Win32	Windows 32-bit user interface
WinLogon	Default logon process in NT

Security flaws in popular security software: Lessons learned from problems in SSH-1.2.17 and older

Antti Huima (antti.huima@hut.fi)*
HELSINKI UNIVERSITY OF TECHNOLOGY

October 9, 1997

Abstract

SSH (Secure SHell) is a nowadays popular security software product, originally created by Tatu Ylönen. In Spring 1997 we began to investigate, in order to find any security shortcomings, the application version **1.2.17** for UNIX^{tm}es. We found several security flaws; they have been fixed in later versions. In this paper, three of these flaws are described in detail. Our purpose is to exhibit how subtle security flaws can exist. We also discuss the underlying reasons of the flaws, trying to reach the roots of the problems in order to give the readers further insights into the area of security engineering. We conclude that trusthworthy security engineering is a hard task and should be performed by specialists.

1 Introduction

SSH (Secure SHell) is a computer application whose purpose is to allow for secure terminal connections via untrusted transport media, for example the Internet. As such it can be seen as a secure replacement of such applications and protocols as `rlogin` [1] and `telnet` [2, 3].

In October 1996, the version number **1.2.17** of the application for UNIXes was released. This release was relatively stable; the next version was not announced before March 1997.

In Spring 1997 we began to investigate the application in detail, trying to find possible security flaws and shortcomings. We examined the most recent version of the application which was, as stated above, the version number 1.2.17. We found various security holes. The finding of these holes was the prominent reason for the new release, which fixed most of the bugs. The most recent software version at the time of writing is 1.2.21.

Our purpose is to explain and examine some of the most interesting flaws we found. We do this in order to gain more insights into the area of security engineering. Especially we want to show how subtle security flaws can exist and how hard they are to spot. After all, the version 1.2.17 and older were used in more than fourty countries by many prominent organisations.

The rest of this paper is structured as follows. In the next section we describe how SSH works in short, and in particular how SSH-1.2.17 *did* work. In Section 3 we describe the nature of the flaws we found in general, including flaws that are not discussed here further. In Section 4 we elaborate on the three flaws we have chosen to present in this paper. The flaws are further discussed in Section 5. Conclusions are drawn in Section 6. The paper concludes with a bibliography.

2 How did SSH-1.2.17 work?

The logical workings of SSH-1.2.17 were presented in the SSH RFC [4]. It was distributed with the software, but after the flaws were found some parts of it were changed; the RFC distributed with the most recent versions of the software differs from the one we refer to.

From now on we refer to SSH-1.2.17 just by "SSH" and write in present tense as "in SSH it is

*We want to thank Tuomas Aura for the remarks he gave during the work and Tatu Ylönen, the original author of SSH, for the patience and interest he expressed towards us during our efforts to find these flaws.

the case, that..." instead of saying "in SSH-1.2.17 it was the case, that...". This must be understood when reading the rest of this article.

We depict here shortly the main workings of SSH. Some of the details will appear in the later sections. We encourage the reader to consult [4] for a complete reference.

SSH is a client-server application. A client (`ssh`) connects to a server (`sshd`). The purpose of the connection is to establish a secured (i.e. secured from eavesdropping and modification) terminal session between the user operating the client and the host running the server.

The client initiates the secured terminal session. It connects to the server using an existing stream-like connection protocol, for example TCP/IP. On the top of this stream SSH runs its own simple packet protocol. Using this protocol the following happens.

First the server sends the client two RSA [5] public keys. The keys are called *the host key* and *the server key*.

The host key is constant during a long time period. It is used to convince the client that it is talking to the correct server. The client compares the received host key to a key it has previously associated with the host it is connecting to. If the keys match, the client knows that only the real host can decrypt anything encrypted with the public key. If the keys mismatch the client gives a warning message to the user and optionally disconnects the session immediately. Automatic disconnection happens if optional *strict host key checking* is used. This is not the default.

The server key is regenerated by default once an hour. After a server key has grown old it is discarded. The purpose of the server key is to ensure old sessions confidentiality even when the static host key gets compromised.[1]

Along the keys the server sends also a [magic] cookie, i.e. a big random number. Both sides compute a new number called *the session identifier* as an MD5 [6] hash of the moduli of the both public keys, and the magic cookie.

The client then generates *a session key*, which is a 256-bit random bit string. The client XORs the session key with the session identifier. Then it encrypts this result with both the RSA keys sent by the server. The client sends this encrypted key along with some parameters to the server. The server is able to recover the session key as it is able to decrypt using the corresponding private RSA keys. By XORing the session identifier with the decrypted result, the original session key—invented by the client—is recovered.

At this point both sides turn encryption on and begin to initiate and then run a terminal session. During the initiation phase the client must authenticate itself. Authentication can be accomplished using UNIX passwords, RSA-based authentication, other methods, and combinations of them. Note that the whole authentication phase takes place over an encrypted channel.

3 Nature and consequences of the flaws found

What we found is that the access control provided by SSH-1.2.17 in the default situation is not theoretically much better than that provided by `rlogin`, the program and protocol that SSH was mainly intended to replace, although in practice SSH is more secure. Active network-level attacks (such as IP spoofing and TCP/IP connection stealing [7, 8]) that render `rlogin` insecure also allow intruders to gain unprivileged access in target systems using only SSH configured in the normal way. However, SSH still provides confidentiality of sessions and integrity to some extent, both properties that `rlogin` lacks; and for example password sniffing does not work with SSH.

We found, though, some attacks against integrity and confidentiality also.

This is not to say that the attacks against SSH were trivial. To find the attacks presented here, much knowledge about the internal workings of SSH, cryptographic protocols and cryptographic algorithms is required. After finding the flaws, the ability to launch active network-level attacks is still needed. Moreover, some of them can be prevented by reconfiguring SSH.

The flaws presented here were relatively easy to fix. Still they are a signal of the fact that the protocol and the implementation designs were not considered carefully enough.

[1] This is called *perfect forward secrecy*.

4 Flaws found

Due to the lack of space, we must be selective about what flaws to present here. We concentrate on the following problems:

Flaw i. In the default configuration, legal RSA authentication can be stolen and forwarded by a malicious party due to a bug in the derivation of the session identifier. This enables intruders to open full-working connections having the rights of a legal user.

Flaw ii. Due to the use of CRC to protect message integrity and the way SSH operates on its data packets, almost arbitrary data packets can be removed, by an intruder, from an active connection between two legal parties.

Flaw iii. In a network where hosts allow the RSARhosts authentication, legal users can use SSH to switch their user identities to what they wish, assuming they have access to the physical network.

4.1 Authentication forwarding

The authentication forwarding flaw is the most severe of the flaws.

We assume that there are a client (C), an intruder (I) and a server host (S). S accepts the RSA authentication from C. I is a potent intruder and can perform active network-level attacks.

4.1.1 RSA authentication

The RSA authentication works as follows. When a client and a server have agreed upon the encryption keys they turn encryption on. After that, all data transmitted between the two hosts is confidential. The client needs next to authenticate itself to the server—as a certain server-side user—in order to gain privileges and get a shell. If the client chooses to use RSA authentication, it tells the server the wanted server-side user name and a public key that it wants to use to authenticate itself. The server checks if a corresponding private key is found from the server-side user's home directory. If it is, the server encrypts a random number, called *the challenge*, using the private key it found. This encrypted challenge is sent back to the client.

Now the client must prove that it possesses the same private key that was found from the server-side user's home directory, thus authorizing itself as the user. If it has the key it succesfully decrypts the challenge. Then it concatenates the decrypted challenge with the session identifier. An MD5 checksum of the resulting data is sent back to the server by the client. The server computes an MD5 checksum of the original challenge in a similar fashion. Assuming that the data sent by the client matches the data computed by the server, the server concludes that the client knows the private key and grants it the user's privileges.

The MD5 hashing is necessary in order to avoid a certain attack against RSA (see e.g. [9]).

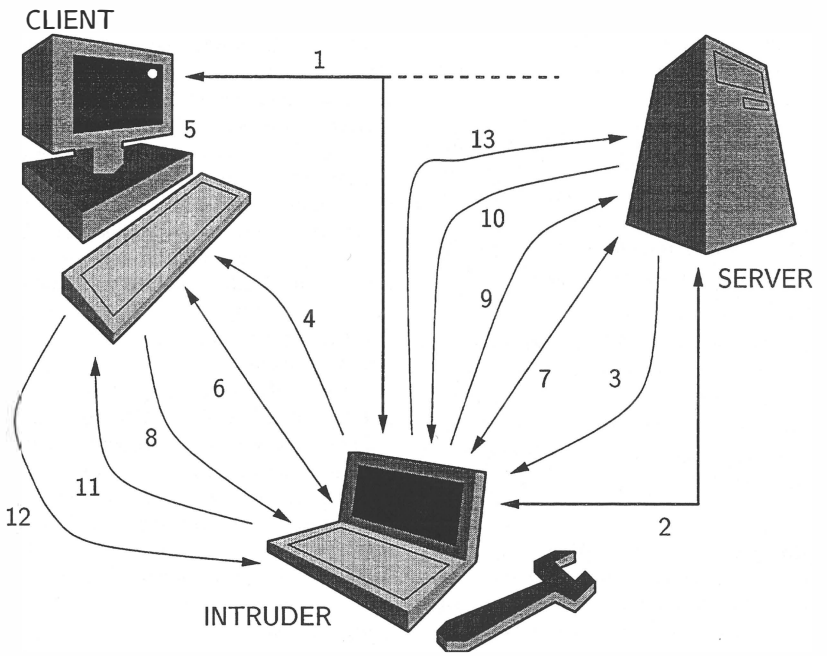
The session identifier is concatenated to the challenge in order to hinder authentication forwarding [4]. The idea is that the response to the same challenge is different in different sessions, and thus a response given in one session is invalid in any other. Basically this is the correct idea.

4.1.2 Flaw

However, there is a flaw. The flaw is that the session identifier does not *really* contain the public keys of a server, *but only the moduli.* What are missing are the encryption exponents. Thus, changing the exponents of the host and server keys does not change the session identifier.

Thus the following attack is possible. C tries to connect to S in order to start an SSH session. I notices this and connects also to S. At the same time I steals the TCP/IP connection between C and S, pretending to be S to C.

I then receives from S the public host and server keys of S, and a cookie as explained in Section 2. The session identifier between I and S is computed from the cookie and the moduli of the two keys. I sends the same cookie and the same keys to C, however changing the exponents of the two keys. Thus, the session identifier between C and I is the same as that between I and S, although the exponents have been changed. If I is able to decrypt the session key encrypted with the keys with changed exponents, sent by the client, I can later forward the authentication response of the client to S. If that really succeeds, I will obtain a full-working



1. I steals the intended connection between C and S.
2. I connects to S.
3. S sends its keys K_h and K_s, and a cookie C_S to I.
4. I changes the exponents of the keys to one and sends the modified keys K'_h and K'_s, and the cookie C_S to C. The session identifier is now the same in both sessions. The common session identifier is denoted by *sessId*.
5. C gets a warning about a changed host key but still continues (the default mode of operation).
6. C sends the encrypted session key $K'_h(K'_s(K_{ses}))$ to I. I is able to recover K_{ses} and builds a working connection between C and itself.
7. I performs key exchange with S and gets a working but still unauthenticated connection.
8. C, assuming I to be S, requests RSA authentication for a server-side user U.
9. I requests RSA authentication for the same user U in the real server.
10. S sends a challenge X encrypted with U's public key K_U to I.
11. I forwards the challenge to C, pretending that I invented the challenge itself.
12. C owns U's private key K_U^-. It computes $K_U^-(K_U(X))$, recovers X and sends MD5(X, *sessId*) to I.
13. I forwards the response to S. It is exactly the response S is waiting for. I is granted U's privileges. At this point I drops the connection between C and itself as it is no longer needed.

Figure 1: The authentication forwarding flaw illustrated

session between I and S where I has authenticated itself falsely with C's authentication.

Note that if C has previously connected to S it knows the real public key of S, including its exponent. C will notice that I has changed it. However, in the default situation, C will give a warning of a changed key but connect however and thus allow for the attack.

Because I can change the exponents freely it can change them to one (1) as well. But in RSA encrypting with exponent one results in no encryption at all. I is able to decrypt anything encrypted with public keys whose exponents have been changed to one—and the attack works. This all is illustrated in Figure 1.

4.2 Dropping packets

Another flaw we present allows an intruder to drop almost arbitrary SSH packets from an active SSH connection despite of any cipher used.

4.2.1 SSH Internal Packet Layout

In order to understand this flaw it is necessary to know the packet layout SSH uses internally. It is presented in Figure 2.

Every packet starts with a four-byte *length field*. This field contains the total length of *the packet payload* and the four-byte CRC *checksum* of the packet, which is found from the end. After the length field come some random bytes, called *the padding*. The length of the padding depends on the value of the length field in the following manner. There are from one to eight bytes padding. When the length of the padding is summed up to the length of the payload and the CRC checksum the total is a multiple of eight bytes. Thus the amount of data after the length field is a multiple of eight bytes. This is to make encryption and decryption using block ciphers easy. For example, DES [10] and IDEA [11] use eight-byte blocks.

The CRC at the end of the packet is a standard 32-bit CRC checksum of the padding and the payload, computed before any encryption has taken place.[2] Note that the length field is excluded from the computation.

[2]SSH uses the polynomial `0xedb88320`.

The payload field contains all useful data of the packet. The field has its own internal data encoding. It is explicitly allowed to contain excess data which must be ignored by the implementations [4].

Note further that the length field is never encrypted. This allows an attacker to be able to find out the start and end positions of any packet in the data stream. However, everything else is encrypted.

4.2.2 Attack

We are now able to describe the attack. It is based on concatenating two consecutive packets in a manner that causes the second packet to be completely ignored.

Namely, assume that two consecutive packets are sent during an SSH session from one host to another. What the attacker does is that he first removes the length field of the second packet. Then he calculates the length of all remaining data in the second packet, and adds the result to the first length field. This is possible because the length field is unencrypted. Moreover, the length field is not used in computing CRC, thus it is not protected from tampering. This process is illustrated in Figure 3.

We want now to show that the resulting packet is a valid SSH packet. First note that L, the length of the second packet without its length field, is a multiple of eight. As a multiple of eight is added to the length of the first packet the amount of assumed padding in the resulting packet is not changed. Thus the padding of the first packet is still correctly skipped over when the packet is parsed.

The CRC checksum of the first packet as well as the padding and payload of the second packet have become excess data added after the payload of the first packet. As it is legal to append whatever excess data to a packet they are fine there. What remains to show is that CRC B is equivalent to the CRC checksum of the resulting packet. But the CRC checksum of the first packet's padding, payload *and* its CRC field is zero due to the properties of CRC checksumming. On the other hand, the CRC checksum of a message does not change if the message is prefixed with data whose checksum is zero. These two facts together cause CRC B to be exactly the same as the CRC checksum of the whole new packet.

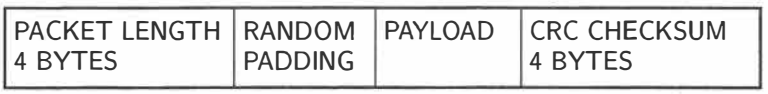


Figure 2: The layout of an internal SSH packet

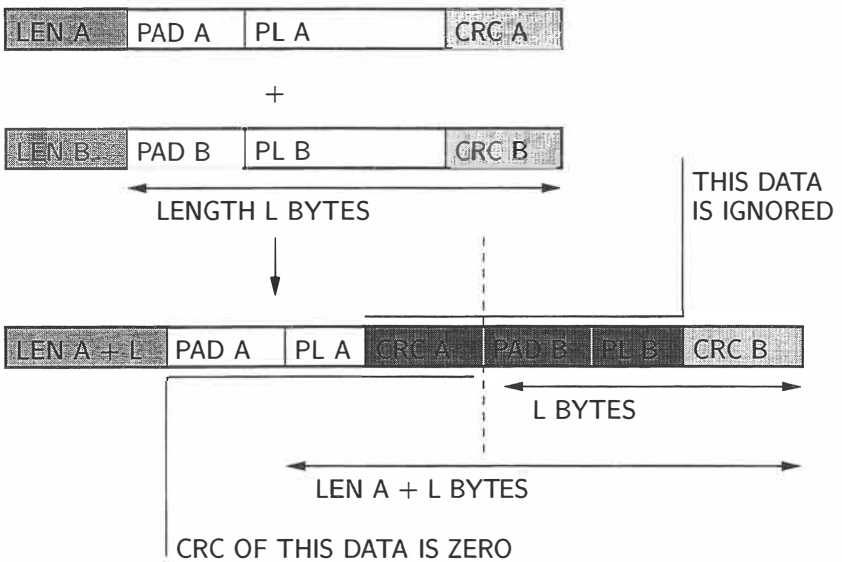


Figure 3: Concatenating two packets in order to logically delete the second

Note further that as no encrypted parts of packets are dropped out the encryption state is kept consistent. The packets have no serial numbering, thus this attack is undetected. SSH provides packet compression as an option. This attack does not work if compression is used.

4.3 Switching user identity by exploiting RSARhosts authentication

4.3.1 RSARhosts authentication method

RSARhosts authentication is an authentication method that is based on authenticating the client host instead of the individual client user. After the host is authenticated using RSA the authenticated client host tells the server the client-side user name. The server host trusts the client host and grants the client the privileges of the server-side user of the same name.

In general, this style of authentication is not applicable because hosts do not trust most of the others. However, in local networks where there is a group of hosts trusting anothers mutually and having the same user data base it can be extremely convenient. The assumption is that if somebody is able to crack one of the hosts he can break into all of them. Thus it makes sense to allow users to wander from one host to another using host-based authentication. Note that this method of authentication does not rely on the (in)security of the TCP/IP layer but uses a strong cryptographic method to authenticate the client.

In situations where RSARhosts authentication is used the SSH client is running setuid-root. This means that a non-root user cannot access his or her own client and thus modify it to send a faked user name. This is the crucial point the method relies on. The client host sends the client-side user name to the server host using the secured communication channel, which should protect the user name from being tampered—including tampering from the [client-side] user himself.

4.3.2 Problem

However, there is a problem. The problem is that the user can still choose the encryption algorithm used between the client host and the server even though RSARhosts authentication is used. *Thus the user can turn the encryption also off.* (The users are allowed to select the usage of no encryption in the default situation.) Having done this, the data going forth and back between the client and the server is completely unprotected. If the user can access the network between the client host and the server he can modify the client-side user name sent and thus gain illegal privileges in the server. This attack is illustrated in Figure 4.

5 Discussion

In this section we want to discuss further the flaws that we presented technically in the previous section.

5.1 The authentication forwarding flaw

The reason behind the authentication forwarding flaw is the improper derivation of the session identifier. As the session identifier is used to protect the client's RSA authentication, the client should be ideally able to ensure that the session identifier is fresh every time. It was assumed that SSH provided a slightly weaker condition; namely that the client were able to ensure that different servers could not force the same session identifiers to be used without using the same host and public keys. However, this assumption is not fulfilled due to a mistake made in binding the session identifier to the RSA public keys.

This flaw was fixed in the subsequent versions by making the client reject any RSA keys with public exponent one. However, this alone does not protect against a less serious variant of this attack: the authentication can be still forwarded if the client chooses to use no encryption as then the server *does not need* to be able to decrypt the session key sent as it is used for nothing.

These flaws are due to a non-obvious mistake that was done in the early development process. Ultimately the protocol should be fixed, although the attacks can be prevented in other ways in practice.

It must also be noted that the traditional protocol development techniques, such as removing redundancy, do not necessarily work in a cryptographic setting. Namely, the modulus of an RSA public key is normally enough to identify an RSA key. The presence of an active attacker however changes the situation here.

5.2 Dropping packets

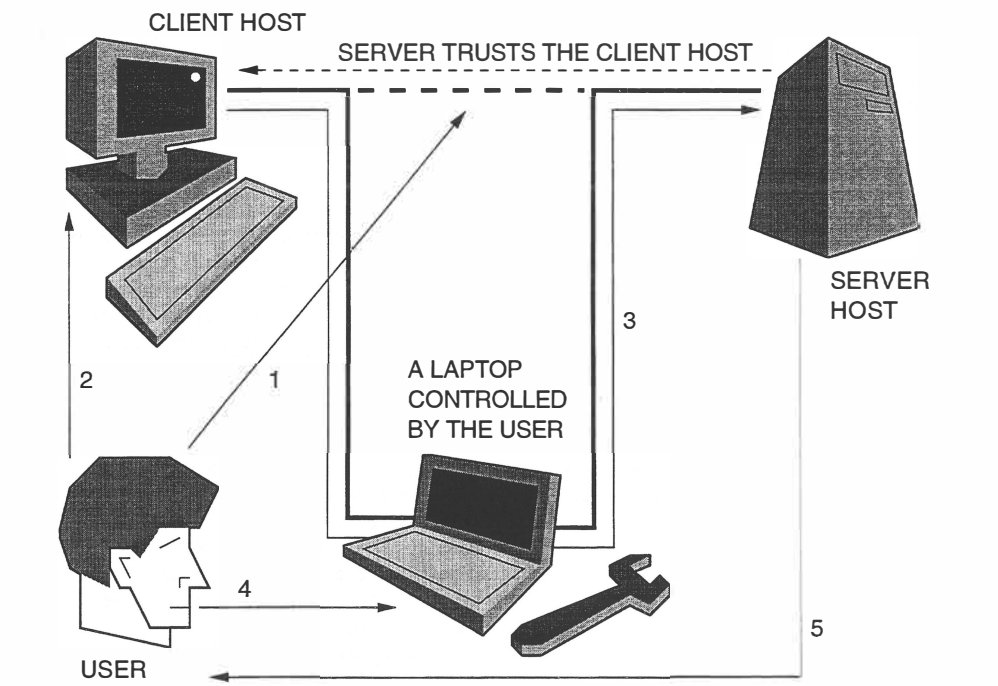
The possibility to drop packets originates from the cryptographically weak message authentication code that is used, namely CRC. CRC was developed to detect random errors but not deliberate tampering of data. With a strong MAC the attack would not be possible. Also using packet sequence numbers would help.

The usage of CRC results in some other flaws not discussed here also.

5.3 Exploiting RSARhosts authentication to gain unprivileged access

The problem is that the server trusts the channel between it and the client, even though it is parametrized by an outsider, namely the user who can turn out to be malicious. In other methods of authentication this is not a problem, because any weakness in the channel turns out to be a loss for the user only. But in this case the user controls the security options of another entity, namely the client host.

A careful analysis of the trust relationships between the server, the client and the user could have revealed the problem. Trust model is different in RSA and RSARhosts authentications. In



1. The user redirects the network connection between the client host and the server host trough a laptop controlled by himself.
2. The user requests the client host to connect to the server, using RSARhosts authentication. The server accepts RSARhosts authentication. The user requests no encryption to be used.
3. The client connects to the server and establishes an insecured communications channel.
4. The user modifies the client-side user name as it passes through the network.
5. The server grants the user (actually the instance of an SSH client in the client host) illegal privileges of another user. The communications channel is still insecured but the user is not disturbed by that for now.

Figure 4: The user identity switching possibility illustrated

RSARhosts authentication the server trusts the following things: (1) the client host is trustworthy; and (2) the client-side user name is transferred to the server using secure methods. However, the user can compromise (2), making the attack possible. (In RSA authentication the server does not need to trust the client nor the transport.)

Why has this now obvious hole remained undetected? Normally disabling encryption is on the risk of the user and any security breakdowns resulting from it harm only the user, not the client or server hosts themselves. Thus disabling encryption was seen as a marginal case during the development process. It was assumed that no user would use that for anything security-critical and thus its impact on the properties of the RSARhosts authentication was obviously left without enough consideration.

5.4 How the flaws were found?

We found the flaws through an intimate understanding of the internal workings of SSH, knowledge of cryptography and some weeks of hard thought. We strongly assume that currently no automatized tools exist that were able to find these flaws. We see this as an exhibit of the fact that real-world security holes are seldom found using mechanical methods. This should be noted with caution.

6 Conclusions

In this paper we discussed some of the flaws that we found in SSH-1.2.17. We first described the flaws and attacks exploiting them in detail, and then considered the reasons that caused the flaws to exist. These flaws were present in the version 1.2.17 of the software and were fixed in the subsequent releases. The flaws were of such nature that the probability of finding them using any established automatic verification methods is low.

The final conclusion we draw is that engineering trustworthy security protocols is a hard task and should be performed with great care by specialists.

References

[1] B. Kantor. BSD rlogin. Internet Request for Comments, December 1991. RFC 1282.

[2] J. Postel and J. Reynolds. Telnet protocol specification. Internet Request for Comments, May 1983. RFC 854.

[3] J. Postel and J. Reynolds. Telnet option specification. Internet Request for Comments, May 1983. RFC 855.

[4] Tatu Ylönen. The SSH (secure shell) remote login protocol, November 1995. The version distributed with the SSH software release 1.2.17. The document went through some changes after the security flaws were found.

[5] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[6] Ronald L. Rivest. The MD5 message-digest algorithm. Internet Request for Comments, April 1992. RFC 1321.

[7] R. T. Morris. A weakness in the 4.2BSD UNIX TCP/IP software. Computing Science Technical Report No. 117, AT&T Bell Laboratories, Murray Hill, New Jersey, 1985.

[8] S. Bellowin. Security problems in the TCP/IP protocol suite. *ACM Computer Communication Review*, 19(2):32–48, April 1989.

[9] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C.* John Wiley & Sons, New York, 2nd edition, 1996.

[10] National Bureau of Standards. Announcing the data encryption standard. Technical Report FIPS Publication 46, National Bureau of Standards, January 1977.

[11] X. Lai and J. Massey. A proposal for a new block encryption standard. In I.B. Damgård, editor, *Proc. EUROCRYPT 90*, pages 389–404. Springer-Verlag, 1990. Lecture Notes in Computer Science No. 473.

An Approach to UNIX Security Logging

Stefan Axelsson, Ulf Lindqvist, Ulf Gustafson, Erland Jonsson

Department of Computer Engineering
Chalmers University of Technology
S-412 96 Göteborg, Sweden
email: {sax, ulfl, ulfg, erland.jonsson}@ce.chalmers.se

Abstract

This paper suggests a very simple and cheap logging policy, light-weight logging, that can easily be implemented on a Unix system, particularly on Sun's Solaris. It is based on logging every invocation of the *exec(2)* system call together with its arguments. We have in the past performed a number of realistic intrusion experiments. We use the data from these experiments to show the benefits of the proposed logging. It is shown that our proposed logging consumes fewer system resources than comparable policies, while still being more effective.

1 Introduction

The main purpose of logging for security reasons is to be able to hold users of the system accountable for their actions [DoD85]. Logging is one of two basic requirements for this to be possible, the other being identification/authentication. It is impossible to hold a particular user accountable for some action indicated in the logs if it is possible, or even plausible, that someone else has "masqueraded" as the user.

Even though less than perfect accountability may result from the mere existence of a log, the logging mechanism serves other useful purposes [NCSC88]:

> It makes it possible to review the patterns of use, of objects, of users, and of security mechanisms in the system and to evaluate the effectiveness of the latter.
>
> It allows the site security officer to discover repeated attempts by users of the system to bypass security mechanisms.
>
> It makes it possible for the site security officer to trail the use (or abuse) that may occur when a user assumes privileges greater than his or her normal ones. While this may not have come about as a result of a security violation, it is possible for the user to abuse his or her privileges in the new role.
>
> The knowledge that there is a mechanism that logs security relevant actions in the system acts as a deterrent to would-be intruders. Of course, for a security logging policy to be effective in a deterring capacity, it must be known to would-be intruders.
>
> The existence of a log makes "after the fact" damage assessment and damage control easier and more effective. This in turn raises user assurance that attempts to bypass security mechanisms will be recorded and discovered. Logs are a vital aid in this aspect of contingency resolution [Kahn95].

In UNIX environments in general, and in the systems under discussion in particular, some of the above mentioned aims cannot be fully realized. For instance, once a user has assumed super-user privileges in a UNIX system, he (or she) then typically has the power to turn off logging, alter existing logs, or subvert the running logging mechanism to make it provide a false record of events. Furthermore, UNIX systems typically do not use sufficiently strong methods of authentication to make it possible to hold a user accountable on the grounds of what appears in an audit trail. In either case, the knowledge that a security violation has taken place is to be much preferred to the situation in which a breach of security has taken place, but gone unnoticed.

The main problem with collecting audit data for a log is not that it is difficult to collect a enough data, but rather that it is altogether too easy to collect an overwhelming amount of it. The sheer volume of the audit data is the immediate reason that logging is often considered a costly security measure. The collection of a large amount of audit data places considerable strain on processing and storage facilities, not to mention the time that must be spent sifting through the logs in order to find any breaches of security. The trade-off is between logging too much, and being drowned in audit data, and logging too little to be able to ascertain whether indeed a breach has taken place.

Most UNIX installations do not run any form of security logging software, mainly because the security logging facilities are expensive in terms of disk storage, processing time, and the cost associated with analysing the audit trail, either manually or by special software. In this paper we suggest a minimal logging policy, light-weight

logging, based on the one single system call *exec(2)*. We use empirical data derived from practical intrusion experiments to show that this logging is sufficient in most cases for tracing the encountered intrusions.

In the following, Section 2describes the suggested policy, Section 3 describes the collection of intrusion data, Section 4 describes the intrusion data, and Section 5 concludes our argument.

2 Light-weight logging

2.1 Definition

We strive for a logging policy that would allow us to detect and trace attacks against our system. Our main purpose is not to be able to detect intrusions in real time, but rather to provide the site security officer with an audit trail from which he can establish exactly what occurred, after the fact. In our view, the logging policy that produces the audit trail from which more detailed information about exactly *how* an intrusion was performed, in addition to just that is *was* performed, is the more successful one. The logging policy should meet the following requirements:

(1) The system should be transparent to the user, i.e. it should behave in the manner to which he has been accustomed.

(2) Since system resources are always sparse, as few as possible should be consumed. This means minimizing the use of storage space, processing time, and administrator time.

(3) While meeting the above requirements, sufficient data should be recorded to maximize our chances to detect and trace any and all intrusions.[1]

We have found that it would be possible to trace *most* of the intrusions presented in this paper by logging relevant information about each *exec(2)* system call made in the system. Since the number of *exec(2)* calls roughly corresponds to the number of commands issued by the user, the amount of audit data should be in the same order as that of *pacct*,[2] while recording more security relevant data than *pacct* does.

Unfortunately one cannot configure the SunOS BSM audit mechanism (see Section 3.2) to generate one record for every command executed, like *pacct*. If one wishes to record every invocation of the *exec(2)* system call, one must audit all the system calls in that audit class, in total 15 different system calls. This may produce more audit data than we care to store and process. Furthermore, the arguments to the *exec(2)* call are not recorded, and that fact reduces the quality of the audit data considerably.[3]

2.2 Example

The example below is a classic UNIX intrusion scenario that can be exploited to gain super-user privileges. This security flaw was present in SunOS 4.1.2, the version on which the first experiment was conducted. In order for the flaw to exist, there must be a shell script somewhere on the system that is *setuid* or *setgid* to someone, i.e. it is run with the privileges of its owner, or group, not its caller. The flaw is exploited by the intruder calling the shell script via a symbolic link, and this results in the intruder gaining access to an interactive command interpreter, henceforth called shell.

Table 1: Penetration scenario.

Step	Shell Command	Comment
1	ln -s /u/vulnerable-file -i	Make link to a *setuid* root shell script.
2	-i	Invoke the shell script as -i.
3	root#	The user now has an interactive root shell.

This flaw comes about as a result of a bug in the UNIX kernel. When the kernel executes the shell script, it first applies the *setuid* bit to the shell and then calls the shell with the filename of the shell script as the first argument. If this filename is "*-i,*" the shell mistakes this for the command line switch to start in interactive mode. In later versions of SunOS, 5.x this problem has been corrected.[4]

---

1. Our intrusion data were collected on the premise that the attackers operated as insiders. In order to log data relevant to tracing intrusions from outsiders, a network security tool such as *Tcp wrapper* could be combined with our suggested logging policy. See [Vene92] for a description of *Tcp wrapper*.
2. See Section 3.2.2 for a more detailed presentation of *pacct,* the UNIX process accounting facility.
3. Both these restrictions have been lifted in SunOS 5.x.

To analyse what needs to be recorded in order to trace this intrusion, we look at the system calls made when exploiting this flaw.

Table 2: System calls.

Step	System calls invoked	Comment
1	fork() execve("/bin/ln", "ln", "/u/vul...", "-i") stat ("-i", 0x9048) = -1 ENOENT (No such file or directory) symlink ("/u/vulnerable-file", "-i") = 0 close (0) = 0 close (1) = 0 close (2) = 0 exit (0) = ?	make link to a *setuid* root shell script
2	fork() execve("-i", "-i", ...) sigblock (0x1) = 0 sigvec (1, 0xf7fff94c, 0xf7fff940) = 0 sigvec (1, 0xf7fff8d4, 0) = 0 sigsetmask (0) = 0x1 sigblock (0x1) = 0 . .	Invoke the shell script as -i The shell starts with some calls to *sigblock sigvec* and *sigsetmask*. The system calls that are executed are then dependent on the input to the shell.
3	root#	

Steps 1) and 2) in Table 2 detail the system calls that are invoked when running *ln(1V)* and *sh(1)*. Both these commands are executed by a shell that performs the *fork(2V)/exec(2)* sequence, which is a prerequisite for all command execution in UNIX.

We outline our suggestion for what information to be included in the audit record in Table 3.

Table 3: The proposed system call logging.

Information recorded for *execve(2)*	Step 1 (ln) (example)	Step 2 (sh) (example)
a record creation time stamp	xxxx1	xxxx2
the real UID	5252	5252
log UID	YY	YY
effective UID	5252	0
real GID	11	11
effective GID	11	11
process ID	1278	1280
parent process ID	1277	1277
filename	"/bin/ln"	"./-i"
current working directory	/u/hack	/u/hack
root directory	/	/
return value	success	success
argument vector to *execve(2V)*	"-s", "/u/vulnerable-file", "-i"	"-i"

Perhaps the only field in Table 3 that merits further comment is the field "log UID." We propose that each user be assigned a unique identifier when he logs into the system. This identifier does not change for the duration of the

4. The filename is no longer passed as the argument to the shell. Instead, the shell is passed a filename on the form */dev/fd/X* where *X* refers to the file descriptor of the already open file. See [Steve92] for an introduction to the */dev/fd* interface.

session, even if the user's real UID changes, as a result of an invocation of the command *su(1V)* for instance. The existence of the "log UID" field makes it easier to trace the commands invoked by each user, although it is not strictly necessary. The same information may be distilled from complete knowledge about the branch on the process tree from the root (login) to the leaf (the current process). The log UID simplifies this task; we have borrowed the concept from C2 auditing [NCSC88, DoD85].

From the above audit records it becomes clear that user *5252* executed a *ln(1V)* command that made a soft link with the name "*-i*" to the shell script, and that the user then invoked the shell script via the link.

If we look in detail at the above, it becomes clear that we need only log the invocations of the *execve(2V)* system call made by this user to trace the intrusion. Since we log the argument vector (argv) to the *execve(2V)* call, we need not log the symlink call separately. As can be seen above, that information is recorded when we log the argument vector to the *ln(1V)* command. We have all the data necessary to trace this specific intrusion back to the user that performed it.

In essence, the proposed logging scheme, creates one audit record per command issued. This also holds true for regular accounting, but there are several differences:

- By logging the start of execution of every command instead of the end of execution, we have a better chance of detecting an ongoing intrusion attempt. This is especially true if we consider long running commands that crack passwords or search the filesystem for example. Furthermore, the command that commences the intrusion is logged. This is far from certain if we delay logging until the command has completed execution, since this may already have turned off auditing etc.
- The most severe security intrusions in UNIX environments are often performed by tricking a *setuid* program into performing some illicit action. By logging both the real and effective UID every time a command is to be run, we can detect many such intrusions.
- Regular accounting logs the first eight characters of every finished command but, since programs can be copied and renamed, this is easy to circumvent. By logging the full path name of every command, together with all arguments, the proposed auditing policy is much more difficult to trick.

3 Collecting reference intrusion data

3.1 The experiment

During the years 1993-1996, we have performed a number of intrusion experiments in UNIX systems [Broc94, Jons96, Jons97, Olov95]. The original goal of these experiments was quantitative modelling of operational security, that is, we tried to find measures for security that would reflect the system's "ability to resist attacks." In order to do so, extensive logging and reporting were enforced and a great deal of data were generated. We believe that these data are also useful for the validation of the logging policy proposed in this paper.

During the experiments a number of students (about 25) were allowed to perform intrusions on a system in operational use for laboratory courses at the Department of Computer Engineering at Chalmers in Sweden. The system consisted of 24 SUN ELC disk-less workstations and a file server, all running SunOS 4.1.2 or SunOS 4.1.3_U1. The system was configured as delivered, with no special security enhancing features [Broc94].

The attackers, who worked in pairs, were given an account on the system - thus, they were "insiders" - and were encouraged to perform as many intrusions as possible. Their activities were limited by a set of rules meant to avoid disturbing other users of the system and to ensure that the experiment was legal. Further details are found in the references above.

3.2 The logging

There were three main classes of accounting in the experiment system that were active:

Connect time accounting is performed by various programs that write records into */var/adm/wtmp*, and */etc/utmp* [Sun90]. Programs such as *login(1)* update the *wtmp(5V)* and *utmp(5V)* files so that we can keep track of who was logged into the system and when he was logged in.

Process accounting is performed by the system kernel. Upon termination of a process, one record per process is written to a file, in this case */var/adm/pacct*. Process accounting's main purpose is to provide the operator of the system with command usage statistics on which to base service charges for use of the system [Sun90].

Error and administrative logging is primarily performed by the *syslogd(8)* daemon [Sun90]. Various system daemons, user programs, or the kernel log abnormal, noteworthy conditions via the *syslog(3)* function. These conditions end up in */var/log/syslog* file on the experiment system.

Another class of logging designed with security in mind is the SunOS BSM (Basic Security Module) logging sub system [Sun90]. This logging facility is said by Sun Microsystems to conform to the requirements laid forth in TCSEC C2, even though the SunOS BSM has not been formally certified according to TCSEC. This logging mechanism was not active in the experiment system.

The system logging and accounting files in the experiment system thus consist of */usr/adm/wtmp*, */etc/utmp*, */var/log/syslog*, */usr/adm/pacct*, and */var/adm/messages*.

3.2.1 Connect time accounting

Various system programs enter records in the */usr/adm/wtmp* and */etc/utmp* files when users log into or out of the system. The purpose of the *utmp(5)* file is to provide information about users currently logged into the system, and the entry for the particular user is cleared when he logs out of the system. The *wtmp(5V)* file is never modified in this manner; instead, when the user logs out, another entry is made containing the time he left the system. The *wtmp(5V)* file thus contains a record of each user as he entered and exited the system.

The *wtmp(5V)* file also contains information indicating when the system was shut down or rebooted and when the *date(1V)* command was used to change the system time.

The *wtmp(5V)* records contain the following information:

- The name of the terminal on which the user logged in.
- The name of the user who logged in.
- The name of the remote host from which the user logged in, if any.
- The time the user logged into or out of the system.

3.2.2 Process accounting by pacct

The process accounting system is, as mentioned before, designed to provide the operator of the system with command usage statistics on which to base service charges for use of the system. The *pacct* system is usually activated by the *accton(8)* when booting the system. When active, the UNIX kernel appends an audit record to the end of the log file, typically */usr/adm/pacct*, on the termination of every process. The audit record contains the following fields:

- Accounting flags; contains information indicating whether *execve(2V)* was ever accomplished and whether the process ever had super-user privileges.
- Exit status.
- Accounting user.
- Accounting group ID.
- Controlling terminal.
- Time of invocation.
- Time spent in user state.
- Time spent in system state.
- Total elapsed time.
- Average memory usage.
- Number of characters transferred.
- Blocks read or written.
- Accounting command name; only the last eight characters of the filename are recorded.

3.2.3 Error and administrative logging

Beside the functions described above, many user and system programs use the logging facility provided by the *syslog* service. At system start-up, the logging daemon *syslogd(8)* is started, and processes can then communicate with *syslogd(8)* via the *syslog(3)* interface.

The messages sent to *syslog(3)* contain a priority argument encoded as a *facility* and a *level* to indicate which entity within the system generated the log entry and the severity of the event that triggered the entry. The *syslog* service is configured to act on the different *facilities* and *levels* by appending the message to the appropriate file, write the message on the system console, notify the system administrator, or send the message via the network to a *syslogd(8)* daemon on another host.

In the experiment system, *syslog* was configured to append all messages to */var/adm/messages* and debug messages from *sendmail(8)* to */var/log/syslog*.

4 Intrusion data

Most of the intrusion techniques encountered during the experiments were previously known by the security community by the time our experimenters employed them to break into our system. We have chosen to extract all successful breaches of security, defined as succeeding in performing an action the experimenter was not normally allowed to perform.

The intrusions are categorized in ten broad classes according to what kind of audit trail they leave. For the sake of brevity, only one typical intrusion in each class is described in detail, while the other intrusions in the class are outlined. For each of the intrusion scenarios, we discuss the possibility of detecting an ongoing attack and of tracing and investigating an intrusion. We also discuss, for each category, whether the intrusion can readily be traced by means of the suggested logging mechanism.

The discussion is summarized under the following headings:

System logging; the logging performed by the kernel and system processes such as *init(8)*.

Application program logging; the logging performed by *application programs*, such as *ftp(1C)*, *su(1V)* etc.

Resource utilization; some attacks result in abnormal load on CPU, disk, network, etc., and this can be monitored (and perhaps detected) in real-time.

Light-weight logging (*execve(2V)*); discussion of the validity of the recorded information related to the suggested logging policy.

4.1 Class C1: Misuse of security-enhancing packages

There are several programs available that help the supervisor of a UNIX system to increase security by testing for known security problems. These programs can of course also be (ab)used by an attacker to learn about existing flaws in the attacked system.

Crack The target system did not enforce password shadowing, so any user was able to read out the encrypted password values and mount a dictionary attack by obtaining and executing the publicly available password guessing program *Crack*.

System logging: The execution of these kinds of programs, which are well-known packages consisting of several subprograms, leaves distinct patterns of multiple entries in the *pacct* file that are easy enough to detect and trace, provided the commands are not renamed.

Application program logging: If the packages are retrieved directly from an Internet archive with *ftp(1C)* or e-mail, this can be traced in */var/log/ftplog* or */var/log/syslog*.

Resource utilization: Possibly massive disk (network if NFS) and CPU utilization.

Suggested logging: The program is recorded and saved with its arguments. Each spawned subprogram in the collection will also be recorded together with its arguments. This approach gives more accurate information considering the patterns in the log file.

COPS Intended to be used by system administrators to find security problems in their UNIX installations, *COPS* is a publicly available package that can also be used by attackers. It consists of a set of programs, each of which tries to find and point out potential security vulnerabilities.

Password generation rules When assigning passwords to other students attending diverse courses at the department, a program that randomly creates 7-character lower-case passwords is used. To make the passwords pronounceable and thus easier to memorize, the program makes every password contain 3 vowels and 4 consonants in a distinct pattern. Unfortunately, it turns out that this pattern severely limits the randomness of user passwords and makes exhaustive search feasible. The attackers compiled a dictionary that satisfied the password generation rules and then ran *Crack*.

Conclusion: Both system logging and our proposed logging policy are capable of detecting the use of the security-enhancing packages encountered during the experiment.

4.2 Class C2: Search for files with misconfigured permissions or *setuid* programs

A general search of the filesystem for files for which the attacker has write permission, or files that are *setuid* to some user is often a step performed by the security packages mentioned in C1. We list it as a separate class since many of our attackers performed such a search when first trying to breach security. These attacks also have in common that they are resource-intensive in terms of (network) disk traffic and may be detected because of this.

Search for files with public write permission The target in this attack was carelessly configured permissions on diverse files in the system, especially system files or user configuration files. Files with public write permissions can be modified by arbitrary users, compromising the integrity of the system. During the experiment we chose not to count general searching as a breach in itself; to regarded the action as a breach, we demanded a detailed description of how to exploit the vulnerable file.

System logging: If the *find(1)* command is used, traces of that can be found in *pacct*. However, since the arguments to the *find(1)* command are not available, it is doubtful whether the security administrator can tell the difference between benign uses of *find(1)* and uses that are consistent with an ongoing intrusion attempt.

Application program logging: N/A.

Resource utilization: Possibly massive disk (network if NFS) and CPU utilization.

Suggested logging: The arguments to *find(1),* together with the command name, are logged making it possible to discover what the attacker is searching for. If the attacker tries to hide the name by making soft or hard links to the find command, the links are also traceable.

Search for *setuid* files Wrongly configured *setuid* files may compromise overall system security, especially *setuid* shell scripts or *setuid* programs with built-in shell escapes. In this case we also demanded a detailed description of how to exploit the vulnerable file.

Conclusion: The suggested logging policy can detect usage of *find(1)* for purposes of searching for files as above. System logging, on the other hand, has a difficult time differentiating between suspect and legitimate uses of *find(1)*.

4.3 Class C3: Attacks during system initialization

As the experiment system was configured it was possible to attack it by halting it during system initialization. The single-user root privileges thus obtained were able to be further exploited to become multi-user root.

Single user boot It was possible to boot the clients from the console to single-user mode. This was possible because */etc/ttytab* was not set up to secure console login. Hence, it was possible to modify an arbitrary filesystem on the client. (Although the clients were disk-less, their root directories were mounted via NFS from the file server.)

System logging: Through */var/adm/wtmp*, */var/adm/pacct* and */var/adm/messages* it is possible to tell when the machine was rebooted after it has come up in multi-user mode again. However, the commands executed in single-user mode are not logged, since logging is not active in single-user mode.

Application program logging: N/A.

Resource utilization: limited.

Suggested logging: It is possible to log the actions performed during single-user mode, but this is not normally done. The single-user mode is viewed as a transient administrative state, employed for administrative duties or system repair. As such, the resources necessary for running the logging mechanism may be unavailable. In our case, all but the root partition was mounted read only, making it difficult to store the log on disk.

Inserting a new account into the /etc/passwd file This is primarily a method to become multi-user root. After becoming single-user root, it is possible to insert a new account in the password file. It is then possible to log into that account when the client comes up in multi-user mode.

***Setuid* command interpreter/program in root filesystem** As single-user root, it is possible to make a copy of a command interpreter (shell), change the owner of the copy to an arbitrary user (for example root) and set its *setuid* flag. Then, when the host has entered multi-user mode, the attacker need only execute the copied shell to take the identity of its assigned owner. This method was primarily used to become multi-user root.

File server intrusion through *setuid* program Although the clients were disk-less, all modifications on the clients root filesystems were also available to the users on the file server. In this way, it was possible to become another user by executing a *setuid* program as described in the preceding intrusion scenario.

Conclusion: Since the system is not fully operational during initialization, it is difficult for both methods to detect, let alone trace, any intrusion attempt. It would be technically difficult to design a logging mechanism that would function under such circumstances, since we are in effect giving away super-user privileges to anyone who happens to walk by. This turns this attack into an "outsider" attack, and we must monitor physical access to the computer room in order to have a chance of catching the intruder.

4.4 Class C4: Exploiting inadvertent read/write permissions of system files

As the experiment system was configured, many critical system files and directories were set up with inadvertently lax access permissions. This made it possible for the attackers to modify critical files to subvert system programs, or, in one case, to read data to which the attackers should not have had access.

YP configuration error NIS (see *yp(3R)*)was installed according to the manual, meaning that it was necessary to initialize */var/yp* before bringing the machine up to multi-user mode. Doing this, and neglecting that an appropriate *umask* is not activated by default in single-user mode, may result in dangerous file permission settings on files created under those circumstances. The NIS server configuration database, */var/yp*, in the target system had permission mode 777, that is, writable and readable for all.

/etc/utmp writable The default on the experiment system was for */etc/utmp* to be writable for all users. This makes it possible for an intruder to hide from appearing in the output of commands such as *who(1)* and *users(1)*, but, more importantly, it allows a user to alter system files. This is accomplished by editing */etc/utmp* and then issuing a *setuid* command (*write(1)* for instance) that uses */etc/utmp* to find its output file.

System logging: *pacct* will show only that some user issued, for instance, a *write(1)* command that looks benign enough. There is certainly no way of differentiating this use of *write(1)* from ordinary legitimate uses of that command.

Application program logging: N/A.

Resource utilization: Limited.

Suggested logging: Unfortunately, if the command the user uses to manipulate */etc/utmp* does not stand out in the audit trail, this kind of intrusion will be difficult to trace. Since we log all arguments to commands issued, our chances of catching the modifying command will have increased substantially in comparison with the chances that *pacct* has.

Crash the X-server When a client wishes to connect to the X-server on the local machine, the client looks for a UNIX domain socket at a predefined location in the filesystem (*/tmp/.X11-unix/X0* in the experiment system). This socket and its directory were world-writable in the experiment system and, as a result, the user could remove the socket and thus hang the X-server. By replacing the socket with a non-empty directory and forcing the X-server to restart, the directory is unlinked, and the files are left "hanging." This would require the supervisor to manually fix the system on the next boot with *fsck(8)*.

Reading of "mounted" backup tapes In the experiment system, backup tapes were present in the tape streamer awaiting that night's backup run. Since the backup tapes were constantly reused (a fairly common policy in many installations) and the tape streamer's device file had world-read permissions set, it was possible for the attackers to read the previous week's backup tapes. This enabled them to read files that they would not normally be allowed to read.

Conclusion: Misuse of system files with erroneous permissions is often detectable by the proposed logging scheme. This misuse often manifests itself as a suspect argument to a user command, that is, the command accesses a file that it should not normally access. System logging has very slim chances of detecting this, since arguments are not logged.

4.5 Class C5: Intercepting data

Due both to configuration errors and the nature of certain UNIX system applications, it was possible for the attackers to intercept communication between users and the experiment system.

Snooping the X-server A publicly available program called *xkey* was used to listen to traffic to the X-server. This program tries to connect to the X-server on the target machine and, if the request is granted, makes it possible to intercept any keystrokes typed by the console user at the target machine.

Frame buffer grabber If a person can log into a SunOS 4.1.x workstation from a remote host, he or she may be able to read the contents of the console's video RAM memory. The frame buffer character special file, */dev/fb*, is per default writable and readable for everyone.

Ethernet snooping Many typical UNIX clients for remote login transmit authentication information in the clear via the network. It is thus possible for someone with access to the network (network topology and technology permitting) to eavesdrop on this traffic and learn passwords, etc. In most UNIX installations one must already have local super-user privileges to be able to perform this kind of attack, and this was the case in the experiment system when the attackers performed the intrusions.

System logging: *pacct* records the commands issued by every user, including root. If the super user has run any of the more popular network listening tools, they should appear in the log entries of *pacct*, that is, unless the intruder has already turned off logging, which unfortunately is very likely.

Application program logging: If a popular package such as *tcpdump* or *esniff* was fetched via *ftp(1C)*, the file */var/log/ftplog* should show traces of that.

Resource utilization: Listening to all network traffic (by setting the network interface in "promiscuous mode") can load the local host heavily.

Suggested logging: The same argument as for *pacct* above applies. However, since the arguments to, for instance *tcpdump,* are recorded, it should be easier to differ between legitimate and illegitimate uses of *tcpdump*.

Conclusion: The snooping that is described above is performed by special programs, most of which are not part of the system distribution. It is easy for the attacker to rename popular packages when installing them, thus foiling our logging effort. In the experiment, most attackers did not bother with this, and hence their intrusions were relatively easy to trace, since these are programs that normally should not be run.

4.6 Class C6: Trojan horses

Some attackers made unsuspecting users execute Trojan horses, that is, applications that purported to do something benign, but did something sinister in addition to that.

Trojan su It is possible for anyone to create a fake *su(1V)* program that when executed saves a copy of the entered password and then prints an error message "su: Sorry" as though the password were wrong. The program then erases itself from the filesystem.

System logging: Normally, when *su(1V)* is executed but the result is unsuccessful, a record starting with *"#su"* is found in *pacct*. The "*#*" indicates that the program was executed with root privileges.[1] Consequently, the Trojan horse *su*, which does not run with root privileges, should appear as a plain *"su,"* which can easily be detected. We cannot judge whether this can be circumvented by a more carefully designed Trojan horse. With regard to tracing, the only thing *pacct* tells us is the user who executed the Trojan *su*, not the location or creator of the program.

Application program logging: N/A.

Resource utilization: limited.

Suggested logging: Enough information to trace this intrusion is recorded since the intrusion method requires that the command is invoked as *su*. Information is recorded on which user executed the program and the full path of the program, and both real UID and *setuid* settings are logged. Since the full path of the program is logged, the chances of discovering who actually planted the fake *su* program increases.

Trojan e-mail attachment This is somewhat of a social engineering attack. One group of attackers sent an e-mail message that contained what was announced to be a picture of explicit nature. However, the picture could not be viewed by normal software already installed on the system, but only with the supplied software, which was also attached to the e-mail. That software was a Trojan horse that hijacked the viewer's account before it correctly displayed the picture.

Conclusion: System logging records only the last part of the path to the command. Because of this it is impossible to differentiate between the running of legitimate commands and their Trojan counterpart. The suggested logging mechanism detects these attacks, since it will show the execution of a normal system command with a suspect path, or the running of a command that has been introduced into the system in a suspicious way.

4.7 Class C7: Forged mail and news

Owing to the design of the mail and news servers on the experiment system, it was fairly easy to send a message that purported to be from someone else.

Faking email On many UNIX systems, it is possible to fake the sender of an e-mail message, making it appear to originate from another user or even from a non-existing user. This is done by connecting to the mail port through *TCP/IP* and interacting directly with the *sendmail(8)* daemon.

1. However, the documentation is unclear as to the exact circumstances under which the "*#*" is inserted by *pacct(2)*. We have found that it is a fairly unreliable indicator as to what privileges were actually acquired by the executing program.

System logging: If the *telnet(1C)* command has been used, we can find a record of that in *pacct* on the sending machine and connect that to the records in *syslog(3)* (described below) through the time stamps.

Application program logging: If the message has been sent from one of our workstations, we can find tracks in */var/log/syslog* left by *sendmail(8)* via *syslogd(8)* on that machine, but only with the faked sender identity.

Resource utilization: limited.

Suggested logging: Clearly, not enough information is recorded to trace the sending of a fake e-mail. Partial information may be available depending on the way in which way the *telnet(1C)* command is invoked or whether the command *mconnect(8)* is used.

Forged news As a consequence of the way in which the remote USENET News server protocol was designed, it is fairly easy to forge a USENET News article to make it appear as though it originated from another user on the system. All authentication must be performed in the news-client software, but nothing prevents a user from connecting to a remote News server by hand, that is, by using *telnet(1C)*.

Conclusion: All attackers that forged mail and news did indeed use the *telnet(1C)* command, passing the parameters on the command line, and were thus easy to detect by the suggested logging, even if the actual mail or news article would be difficult to trace. However, it is trivial to invoke the *telnet(1C)* command without any command line arguments and foil both methods of logging.

4.8 Class C8: Subverting *setuid* root applications into reading or writing system files

Setuid root applications on UNIX systems are allowed to read or write to any file by default. It is imperative that such applications check user supplied arguments carefully, lest they be tricked into doing something that the user would not normally be allowed to do. However, many such applications contain flaws that allow an attacker to perform such unauthorized reading or writing of critical files.

Xterm logfile bug, version 1 The *xterm(1)* client has an option for the entire session to be logged to a file. Furthermore, *xterm(1)* is *setuid* root for it to be able to change the owner of its device file to that of the current user. A bug makes it possible for any user to specify an existing file as the logfile to *xterm(1)* and have *xterm(1)* append data supplied by the attacker to that file.

System logging: From the *pacct* file, all we can see is that someone has run *xterm(1)*, a so common occurrence that we can safely say that it is impossible to trace an intrusion this way.

Application program logging: N/A

Resource utilization: Limited.

Suggested logging: Since we log all the arguments to *xterm(1)* as it is being run, we catch both the invocation of the logfile mechanism and the telltale argument *-e echo "rootː:0:1::/bin/sh"* or a similar one, which is the hallmark of this intrusion.

Xterm logfile bug, version 2 A variation of the preceding exploit, where the attacker can create a file and have the output from *xterm(1)* inserted in that file, provided that the file does not already exist.

Change files through mail alias The UNIX operating system maintains a global mail aliases database used by the *sendmail(8)* program to re-route electronic mail. One standard alias delivered with some versions of UNIX is *decode*. By allowing this alias, it is possible for anyone to modify some directories in the system.

Finger daemon The *finger(1)* utility in the experiment system is *setuid* to root, and it makes an insufficient access check when returning information about a user. It is possible to make finger display the contents of any file via the use of a strategic symbolic link from *.plan* in the user's home directory, to the target file.

Ex/vi-preserve changes file The venerable UNIX text editors *ex(1)* and *vi(1)* have a feature by which in the event that the computer crashes or the user is unexpectedly logged out, the system preserves the file the user was last editing. The utility that accomplishes this, *expreserve(8)*, is set UID to root, and has a weakness by which the attacker can replace and change the owner of any file on the system.

/dev/audio denial of service Owing to a kernel bug, it is possible to crash the machine by sending a file via *rcp(1C)* to */dev/audio* on a remote machine.

Interactive *setuid* shell This problem in SunOS 4.1.x has been fixed in more recent operating systems. An attacker simply invokes a *setuid* (or *setgid* or both) shell script through a symbolic link, which immediately results in an interactive shell with the effective UID (and/or GID) of the owner of the shell script. This is detailed in Section 2.2.

Conclusion: The methods used to trick the *setuid* program inevitably either supply suspect arguments to the command or modify the filesystem in advance to running the *setuid* command. Depending on the specific circumstances, our logging proposal has a high chance of detecting and tracing the intrusion, since we either log the suspect arguments themselves or log the commands that poison the filesystem. System logging does not manage to accomplish either.

4.9 Class C9: Buffer overrun

As mentioned above, a *setuid* program must be careful in checking its arguments. There exists a class of security flaws where the attacker subverts the *setuid* application by filling an internal (argument) buffer so that it overflows into the *setuid* program's execution context. In this way it is possible for the attacker to force the *setuid* program to execute arbitrary instructions. We have encountered only one such attack. We include it here because of the severity of this type of attack and because it is widespread and commonly encountered in the field.

Buffer overrun in rdist The *rdist(1)* utility has a fixed length buffer that can be filled and thus made to overflow onto the stack of *rdist(1)*, which is set UID root. The attackers did not manage to exploit this other than to crash *rdist(1),* but we include it here in any case, as it's an interesting class of intrusions.

System logging: As usual, *pacct* does not manage to leave any conclusive traces in the log files. The invocation of the program with the overflow condition would probably not show up, since the name of the finished program is recorded at the end of execution when the original program image has typically already been overlaid with that of a set UID shell.

Application program logging: N/A.

Resource utilization: Limited.

Suggested logging: Since we log the arguments to the command, *rdist(1)* in this case, we immediately see that something is wrong. We in fact record in the log the entire program that *rdist(1)* is lured into overflowing onto its stack! (We may not always be so fortunate; some variations of these exploits keep the actual overflow code in an environment variable, which we will not log.) Since we see both the original *exec(2)* of *rdist(1)*, followed by the *exec(2)* of a root shell, without the intervening *fork(2V),* we have conclusive proof that the system has been subverted.

Conclusion: See the discussion concerning the above exploit, since it is typical of these kinds of exploits.

4.10 Class C10: Execution of prepacked exploit scripts

It is possible for the novice attacker to download prepacked programs or command scripts that exploit a known security flaw. Such packages are in wide circulation and provide the attacker with an easy entry into the system.

Loadmodule *setuid* root script We have included a specific example of a breach that involves a *setuid* script since this *setuid* script is included in the SunOS distribution tapes. Hence, this security flaw is widespread and, as such exploit scripts have been developed, and widely circulated. This particular exploit script (*load.root*) was found by many experimenters who used it successfully, some without any deeper understanding of the security flaw involved. The exploit script in question uses the environment variable *IFS* to trick *loadmodule(8)* into producing an interactive shell with super-user privileges.

System logging: Since *pacct* does not record the arguments to the commands that the user executes, it is difficult to establish that the *load.root* exploit script has indeed been run. However, as the exploit script executes a number of commands in a predefined sequence, it should, at least in theory, be possible to ascertain that the script has been run with some level of certainty. This is generally not possible without detailed knowledge about the script. As mentioned previously, the "#" marker to indicate that super-user privileges has been used does not appear in this case.

Application program logging: N/A.

Resource utilization: Limited.

Suggested logging: Since we can determine what commands were run and with what arguments, it becomes much easier to determine that a breach has occurred. In particular, the fact that the user has executed a script that in turn invokes *exec("/bin/sh", "sh", "-i")* with root privileges gives the game away. However, the flaw is exploited by the introduction of an environment variable, something which we do not record because this would lead to the storage of too much data. A strong indication that some sort of IFS manipulation has taken place, however, is the fact that the audit trail shows that a user has executed a command named *bin* as part of a shell script.

Mailrace (sendmail) The original mail handling client */bin/mail(1)* is set UID root in SunOS. If a recipient of a mail message lacks a mail box, the program creates one before it appends the mail message to it.

Unfortunately, there exists a race condition between the creation of the mail box and the opening of it for writing. Exploit scripts have been published that exploit this race condition. These operate by replacing the newly created mail box with a symbolic link to any file which, as a result, will be overwritten or created with the contents specified by the attacker.

Conclusion: If the script is known beforehand, its use can be established with normal system logging. The suggested logging makes it a good deal more likely that a script that has not been previously examined can be traced and analysed to determine its effects.

4.11 Summary of results

The results are summarised in Table 4. There were a total of 30 different attacks in 10 classes. Our proposed logging policy does quite well, managing to detect 21 attacks in 7 classes. System logging did not fair as well, only 7 intrusions in 3 classes were detected. As can be seen, neither of the methods succeeded in detecting the 9 attacks in the three classes C3, C5, and C7.

Table 4: Summary of traceable intrusions.

Class (number of intrusions in class)	Suggested logging	System logging
C1: Misuse of security enhancing packages (3)	X	X
C2: Search for files with misconfigured permissions or *setuid* programs (2)	X	
C3: Attacks during system initialization (4)		
C4: Exploiting inadvertent read/write permissions of system files (4)	X	
C5: Intercepting data (3)		
C6: Trojan horses (2)	X	X
C7: Forged mail and news (2)		
C8: Subverting *setuid* root applications into reading or writing system files (7)	X	
C9: Buffer overrun (1)	X	
C10: Execution of prepacked exploit scripts (2)	X	X
Number of classes:	7	3
Number of intrusions (30):	21	7

5 Conclusion

After studying the above security breaches it becomes clear that *pacct* suffers from three major shortcomings:

(1) It does not commit the executed command to the audit trail until it has finished executing. This misses crucial long running commands. Commands, the process image of which is overlaid by a call to *exec(2)*, does not appear in the audit trail, and the command that crashes or compromises the system is at risk of not being included in the audit trail.

(2) *Pacct* does not record the arguments to issued commands. This more often than not turns the log material into a useless alphabet soup from a security perspective, since it is impossible to see what executed commands were set to act upon in terms of files etc. In most of the cases above, the arguments to the commands are what set legitimate uses of the commands apart from illegitimate ones.

(3) The mechanism that is supposed to trace uses of super-user privileges is unreliable at best and downright erroneous at worst. It can thus not be trusted to provide worthwhile information about the use of super-user privileges.

By correcting these shortcomings in the original *pacct* policy, our proposed mechanism manages to trace an overwhelming portion of the intrusions, namely those that fall into the following three categories:

(1) The user has run commands he should not run; the log shows that someone with a log-UID that does not correspond to a known supervisor has executed commands with super-user privileges.

(2) The user has run commands with suspect arguments and, by doing so, has managed to trick a system application into doing something illicit.

(3) The user has run a suspicious-looking sequence of commands. This indicates that the user has run some exploit script or some security enhancing package that he should not have run.

By also logging more security-relevant information pertaining to who executed the command, and the general environment in which it was executed, we have a sufficiently complete record of events on which base further actions, provided that the audit trail is protected from manipulation. This can for example be accomplished by logging to a dedicated network loghost, or to write-once media, as detailed in [Garf96].

Our proposed security logging policy can very easily be implemented using the SunOS BSM module in newer versions of the SunOS operating system [Sun95]. It traces the overwhelming majority of intrusions encountered during our experiments. Since it does not consume much resources in terms of processing power and storage capacity, it can be left running on all machines in an installation. Thus, it can easily be used by anyone as a "poor man's logging."

6 References

[Bish95] Matt Bishop. "A Standard Audit Trail Format." *In Proceedings of the 18th National Information Systems Security Conference*, pages 136-145, Baltimore, Maryland, USA, October 10-13, 1995.

[Broc94] Sarah Brocklehurst, Bev Littlewood, Tomas Olovsson, and Erland Jonsson. "On measurement of operational security." In *Proceedings of the Ninth Annual Conference on Computer Assurance (COMPASS '94)*, pages 257-266, Gaithersburg, Maryland, USA, June 27-July 1, 1994.

[DoD85] U.S. Department of Defense. Trusted Computer System Evaluation Criteria, December 1985. DoD 5200.28-STD.

[Garf96] Simson Garfinkel and Gene Spafford. "Practical UNIX & Internet" Security. O'Reilly & Associates, second edition, 1996.

[Jons97] Erland Jonsson and Tomas Olovsson. "A Quantitative Model of the Security Intrusion Process Based on Attacker behaviour." In *IEEE Transactions on Software Engineering*, vol. 23, No. 4, April 1997.

[Jons96] Erland Jonsson and Tomas Olovsson. "An empirical model of the security intrusion process." *In Proceedings of the Eleventh Annual Conference on Computer Assurance (COMPASS '96)*, pages 176-186, Gaithersburg, Maryland, USA, June 17-21, 1996. IEEE.

[Kahn95] Jay J. Kahn and Marshall D. Abrams. "Contingency Planning: What to do when bad things happen to good systems." In *Proceedings of the 18th National Information Systems Security Conference*, pages 470-479, Baltimore, Maryland, USA, October 10-13, 1995.

[NCSC88] National Computer Security Center. "A Guide to Understanding Audit in Trusted Systems." 1 June 1988. NCSC-TG-001, Version-2.

[Olov95] Tomas Olovsson, Erland Jonsson, Sarah Brocklehurst, and Bev Littlewood. "Towards operational measures of computer security: Experimentation and modelling." In Brian Randell et al., editors, *Predictably Dependable Computing Systems*, ESPRIT Basic Research Series, chapter VIII. Springer-Verlag, 1995.

[Steve92] W. Richard Stevens. "Advanced Programming in the UNIX Environment." Addison-Wesley Publishing Company Inc., 1992, ISBN 0-201-56317-7.

[Sun90] Sun Microsystems. System and Network Administration, March 27, 1990. Part No: 800-4764-10, Revision A.

[Sun95] Sun Microsystems Inc. "SunSHIELD Basic Security Module Guide." Sun Microsystems Inc. 2550 Garcia Avenue, Mountain View, California 94943- 1100 USA.

[Vene92] Wietse Venema. "TCP WRAPPER: Network monitoring, access control and booby traps." In *Proceedings of the 3rd USENIX UNIX Security Symposium*, pages 85-92, Baltimore, Maryland, USA, September 14-17, 1992. USENIX Association.

ISBN 951-22-3783-0
ISSN 1455-0733

LIBELLA PAINOPALVELU OY