General Purpose Operating System for Security-critical Applications

Jaroslav Janáček[*]
Department of Computer Science
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava
Mlynská dolina, 842 48 Bratislava, Slovakia
janacek@dcs.fmph.uniba.sk

Abstract

Computers are used by the general public for an increasing number of tasks where security is an important aspect. Most of the computers are used to execute potentially malicious applications, often without the user's knowledge, alongside applications that process sensitive data. The common security functions of the current common operating systems do not provide sufficient protection of the confidentiality and the integrity of the sensitive data processed by one application against other applications running on behalf of the same user.

We show that the operating system has an important role in security – it is often impossible to effectively deal with security at the application layer without a suitable support of the operating system. We analyze several typical examples of applications used in home and office environments and generalize the security requirements and the security properties of the data used by the applications. We design a security model including a formally defined information flow policy to protect the confidentiality and the integrity of the data. We state and prove basic security properties of the model. We analyze several existing protection profiles for operating systems, and we argue that none of them is suitable for an operating system with the intended use. We present a new protection profile that supports our new security model. Finally, we show that it is feasible to modify Linux operating system to make it compliant with the presented protection profile.

[*]Recommended by thesis supervisor:
Assoc. Prof. Daniel Olejár
Defended at Faculty of Mathematics, Physics and Informatics, Comenius University in Bratislava on November 22, 2010.

Janáček, J. General Purpose Operating System for Security-critical Applications. Information Sciences and Technologies Bulletin of the ACM Slovakia, Vol. 2, No. 2 (2010) 49-59

Categories and Subject Descriptors

D.4.6 [**Operating Systems**]: Security and Protection—*access controls, information flow controls*

Keywords

operating system, information security, security model, information flow policy, protection profile

1. Introduction

During the past few years we have been observing a significant increase in the number of people who use computers to perform tasks where security is important. Typical such applications, that are of interest to general public and can be expected to be used on home computers, include Internet banking, e-government applications, electronic signature creation and verification applications. We will use the term *security-critical application* to denote such applications.

Organizations use information systems to store and process confidential business data and personal data. Unauthorized access to information stored or processed by all of the mentioned applications (and many others) can often cause a substantial loss to the affected person or organization. It is usually the user's responsibility to protect the sensitive data. However, common users are not information security experts and can only follow some guidelines given to them. Even that is usually possible only if the guidelines are simple enough. While a larger organization can dedicate some computers to security-critical applications and protect them against unauthorized access, modification or software installation, it can hardly be expected in a home environment.

In our previous research [12] we have shown that it is generally infeasible to deal with security on the application layer without a suitable support from the operating system. We have also analyzed several common general purpose operating systems and discussed the insufficiency of their security functions to counter some common threats. The principal results of the analysis, in an updated form, are presented in our thesis. It has motivated our work on designing a security model to be implemented in an operating system providing sufficient security functions to allow using a single computer with a single operating system for both security-critical applications and untrusted applications.

2. Security in Common Operating Systems

In the thesis, we have analyzed two groups of currently common desktop operating systems for personal computers. The first group consists of Microsoft Windows 2000/ XP Professional/Vista, and the second one consists of the Linux operating systems.[1].

We have used the security functional requirements classes specified in the international standard ISO/IEC 15408 [4] to organize the analysis of the security functions of the considered operating systems. Operating systems in both groups use hardware resources to protect themselves against manipulation by processes, and to protect the processes against each other. They implement security functions for security audit records generation and review, user data protection, user identification and authentication, security management, limited trusted path and trusted channels, and other security requirements. As far as access control is concerned, operating systems in both groups implement discretionary access control mechanisms, and some of them (Linux, Windows Vista) optionally provide different sorts of (partial) mandatory access control mechanisms.

2.1 Security Problems of Common Operating Systems

We have identified several common security problems that are not addressed sufficiently by the considered operating systems if potentially malicious applications are to be used simultaneusly with security-critical ones.

2.1.1 Abuse of privileges by an administrator

Linux operating systems not not apply discretionary access control to the processes running on behalf of the user with ID 0 (root). A system administrator can, by default, run any program on behalf of this user. The administrator can, therefore, manipulate with data of any user, and can also modify parts of the operating system and applications, e.g. to capture sensitive input such as passwords, PINs, ... The SELinux security module can be used with a carefully specified security policy and with a suitable separation of duty among more people to lower the risks of abuse of privileges by an administrator.

In Windows operating systems, the users can limit the access of administrators to their data. However, this feature is insufficient because it can be circumvented by abusing the privileges for backup operators. The protection of confidentiality of the data can be improved using encryption, but the administrators are still able to overcome the encryption by designating a special user able to decrypt all files.

To sum up, the system administrators (and other users with special privileges) effectively control the entire system. Their trustworthiness is therefore very important for the overall security of the operating systems.

[1]To be correct, there are many Linux distributions, such as Slackware, Debian, RedHat, Suse, Ubuntu, ... A Linux distribution consists of the Linux kernel and a collection of libraries, system utilities and application programs. Its security functions are determined mainly by the kernel itself and by the basic system utilities and libraries that are shared among the individual distributions. This allows us to neglect the differences among the individual distributions and talk about the *Linux operating system.*

2.1.2 Too many processes with high privileges

Another common problem is that many processes providing various services run on behalf of privileged users (e.g. root on Linux, or special system user on Windows). Many of these processes do need the privileges, but there are also many of them that need only a small subset of the privileges. Programs contain various flaws that may allow attackers to execute arbitrary code with the privileges of the exploited process. When such flawed program runs within a process with an administrator's privileges, it may allow an unidentified attacker to abuse the administrator's privileges.

2.1.3 Abuse of privileges of an ordinary user

A currently very common problem is the abuse of privileges of an ordinary user by malicious applications. Many users do not realize that the programs they have obtained from untrusted sources (such as the huge number of web pages) can, on top of (or instead of) the declared activities, perform any operations, including malicious ones, while abusing the user's privileges.

Another problem is that applications often contain flaws that allow a code embedded in a specially crafted document to be executed as a result of the application's processing of the document. When the user uses such application to process a document obtained from an untrusted source, the risk is similar to that of directly running a program from an untrusted source. This problem becomes even more significant when we consider common web browsers or e-mail clients that, to be more user-friendly, automatically run various applications to process documents in web pages or e-mail attachments, often without asking the user whether or not to do so. This is probably the most common way of spreading computer viruses and Trojans nowadays. The lack of awareness of users also helps the attackers to use this way to abuse the users' privileges. The users often trust unauthenticated information. A typical example is the e-mail address of the sender of an e-mail message – it is trivial to forge while many users, seeing a known e-mail address, believe that the message must have been received from a person they trust. The message often comes from a virus, Trojan or another malware that attempts to exploit a security flaw in an application the recipient is expected to use to process the message.

2.1.4 Direct manipulation with the hardware

Direct manipulation with the hardware while it is not controlled by the operating system also presents a nonnegligible possibility of breaking security of the considered operating systems. If an attacker can physically manipulate with a computer, he or she can modify the contents of its hard disks, and thus modify any data, applications or parts of the operating system. Even though the operating systems support digital signature verification for system components or for software packages during installation, the signature verification depends on data stored on the disk that may be subject to unauthorized modification via direct hardware manipulation. The system can also be modified not to perform the verification at all. This problem cannot be, in general, solved at the operating system layer without booting the system from a medium that can be trusted not to have been modified in an unauthorized way (e.g. a physically protected read-only medium or a file the authenticity of which is verified before using it by the computer's firmware and hardware).

2.2 Existing partial solutions

Some of the problems mentioned above can be partially solved using the features provided by the considered operating systems. The problem with processes with too high privileges can, in some cases, be solved by minimizing the privileges to the minimal required set. It is, however, not always sufficient or possible.

The problem with abusing the user's privileges can be sometimes solved by increasing the awareness of the users, and by strict separation of the security-critical activities from the risky ones (such as web browsing or e-mail processing). When a user uses different accounts for different purposes, he or she can set the access control lists in the way that the programs running with the user's identity for the risky operations cannot interfere with the sensitive data that are to be accessed only by processes running with the other user's identity.

There have been many projects for the Linux operating systems (e.g. [6, 8, 9, 1, 7]) implementing various security mechanisms (most often a sort of mandatory access control) to effectively solve some of the problems. They finally resulted in the inclusion of Linux Security Module (LSM) framework in the Linux kernel, and in the acceptance of SELinux module as a standard part of the kernel. SELinux can be used to protect confidentiality using Bell-LaPadula based multi-level security policy, and using any policy specified in terms of domain and type enforcement mechanism. The latter is nowadays used by several Linux distributions (e.g. RedHat, Debian) to limit the impact of exploiting flaws in applications on Linux servers. Attempts to use a strict SELinux policy on desktop systems have failed due to too diverse requirements of desktop systems [11, 2], and they have resulted in usage of so called *targeted* policy that constrains many of the system services and server processes but leaves the user-started processes unconfined in a single domain. This way the user's data are not protected against malicious code started by the user, either directly or indirectly via a flawed application and a malicious document.

Windows operating systems also brought several interesting attempts to solve the problems. Windows Vista introduces two new security features that are worth mentioning. One of them is called Mandatory integrity control (MIC)[18]. Filesystem objects and processes are assigned integrity levels, and a process can only modify objects with the same or lower integrity level than the process's integrity level. In fact, one half of the standard Biba model[20] rules are used. It can also be configured to enforce one half of the standard Bell-LaPadula model[10] – a process can only read from objects with the same or lower level. Because MIC only implements a half of the standard rules, it lacks the provable security properties of Bell-LaPadula and Biba models. Its primary use was to protect the user's data and programs against a malicious code executed by a flawed web browser. It does not, however, prevent other processes from reading (and acting upon) malicious data that have been downloaded from untrusted sources.

Another new security feature introduced in Windows Vista is *User Account Control* (UAC)[19]. It deals with the, well known and unfortunate, fact that many users on desktop systems use accounts with administrator privileges (either to overcome problems with some software or just out of a sort of laziness) for their normal computer usage. This leads to a situation that even a flawed web browser or e-mail client can perform operations that are restricted to administrators, and it effectively makes the access control mechanisms ineffective. UAC works by disabling some of the special privileges normally given to administrators and prompting the user for a permission to grant these privileges to the process that attempts to perform an operation that requires the privileges. This way, the processes cannot perform the privileged operations without the user knowing about it. They can still, however, perform any operations that do not require the privileges dedicated to administrators.

2.3 Existing Protection Profiles

The security requirements for IT products are often specified in the form of a protection profile (PP) according to the international standard ISO/IEC 15408 [3]. In the thesis, we have analyzed several existing protection profiles for operating systems. In the PP registry[2], several protection profiles for operating systems are registered. Two of them, Controlled access protection profile (CAPP)[16] and Labeled security protection profile (LSPP)[17], are for general use. A few others are specifically tailored to the needs of the Department of Defense of the U.S.A. for the classified information processing, but they are not suitable for general home/office use.

The security objectives and the functional security requirements of the CAPP do not cover protection against abuse of an administrator's privileges – a trustworthy administrator is assumed. No protection against malicious code executing with a user's privileges is provided because all access control decisions are based on the user's identity regardless of the program being executed. The user has thus no way of preventing a malicious program from accessing any data accessible to the user. Even if the user attempts to restrict his/her own access rights to an object, the malicious program running on the user's behalf can grant the access rights to the user (or to any other user).

The LSPP improves the protection of confidentiality of data in the environments where information is/may be classified in the Bell-LaPadula way. When a process (subject) is executed at a security level, it cannot read from objects with a higher security level (thus containing more sensitive information), and it cannot write to objects with a lower security level. For example, if communication objects connecting to untrusted external systems are classified at the lowest level, a malicious program running at a higher level cannot send sensitive information to the external systems, and a malicious process operating at the lowest level (and thus able to communicate with the external system) can read no sensitive information (contained in an object with a higher level).

The LSPP, however, contains no improvements regarding integrity protection. A malicious program running at the lowest security level can still cause damage to valuable data stored on the system. As we will show later, the integrity protection can be of equal, or sometimes even higher importance that the confidentiality protection.

The LSPP, just like the CAPP, does not cover the pro-

[2]`http://www.commoncriteriaportal.org/pp.html`

tection against abuse of an administrator's privileges – it assumes a trustworthy administrator.

We can conclude, that an operating system compliant with any or all of the mentioned protection profiles can still suffer from the problems discussed in this section.

3. Goals, Objectives and Methods

The problems mentioned in the previous section have given rise to the primary goals of the thesis:

- designing a suitable security model for an operating system supporting secure use of security-critical applications alongside untrusted and potentially malicious applications,
- creating a protection profile, compliant to the ISO/IEC 15408 standard[3, 4, 5], for a general purpose operating system supporting such use, utilizing the security model.

In order to achieve the goals, we have set the following objectives to fulfil in the thesis:

1. Identify and categorize typical applications and identify the protection requirements.
2. Specify the data classification scheme.
3. Specify the security model.
4. Formulate and prove security properties of the model.
5. Create the protection profile.

In order to specify the security model we have used a simplified model of an operating system consisting of active entities – *subjects* (processes) performing *operations* on passive entities – *objects* (files, directories, communication objects, processes, . . .). We started with read, write, create and delete operations, and we extended the set of operations later to cover a more realistic operating system. We have modelled access control and information flow control using logical functions operating on subjects and objects and yielding true or false depending on whether the operation is permitted or not.

The structure of a protection profile is specified by the international standard ISO/IEC 15408 [3]. We have used the standard to write our protection profile.

4. Our Results

4.1 Classes of Applications Considered

We have considered the typical applications used on personal computers in the home and small office environment. We have identified several classes according to the security requirements for their data.

4.1.1 Malicious applications

A special class of applications is the class of malicious applications. These are applications that have been intentionally programmed to perform malicious activities. The typical examples are computer viruses, worms, Trojan horses and other kinds of so called malware. They can be downloaded from the Internet by the user, received as an attachment of an e-mail message, or a vulnerable application may be turned into a malicious one by processing malicious data. The user is usually unaware of the fact that a particular application is malicious.

It has to be assumed that the malicious applications do anything not prevented by the operating system or the environment of the computer (e.g. a network firewall).

4.1.2 Local applications

The class of local applications contains the applications that are used to process data stored in a local filesystem. These applications generally do not need network access to perform their tasks. The typical examples are text processors, spreadsheets, presentation software, graphic editors, ...

Local applications are used to process data with varying requirements regarding the confidentiality and integrity protection. If they process malicious data, they may become malicious due to programming errors.

4.1.3 Sensitive web access

Web browsers are often used to access remote services that process data requiring confidentiality and/or integrity protection. A typical example is an Internet-banking system. It provides access to financial information; it allows the user to submit transaction orders to the bank, etc. It also processes authentication data (e.g. passwords). All such data may be considered confidential by the user, and therefore, are to be adequately protected. The confidentiality and the integrity of the data during their transmission is usually protected by means of cryptography. Cryptography is usually also used to provide authentication of the remote system. But the data is also to be protected while stored in the memory or in a file on the local computer. Consider an instance of a web browser used for general Internet access. It may have processed some malicious data, a and therefore, it may have become a malicious application exporting everything to an attacker. If the instance of the web browser is later used to access an Internet banking system, all the confidential information may leak.

4.1.4 Digital signature creation

A digital signature creation application needs access to the private key. The private key is a very sensitive piece of information the confidentiality of which has to be protected. The integrity of the private key has to be protected as well because its modification can lead not only to the loss of ability to create correct digital signatures, but also to the leak of information that is sufficient to compute the corresponding private key in certain cases.

4.1.5 Digital signature verification

A digital signature verification application needs access to the public key. The public key requires no confidentiality protection, but it does require integrity protection. If attackers were able to modify the public key used to verify a digital signature, they would be able to create a digitally signed document that would pass the signature verification process.

4.1.6 Data encryption

A data encryption application using asymmetric cryptography needs access to the public key of the receiver of the data. As mentioned above, the public key requires no confidentiality protection, but it does require integrity protection. In the case of encryption, if the public key were modified by an attacker, the attacker would be able to decrypt the encrypted data instead of the intended receiver.

The encrypted output of a data encryption application may be transmitted via communication channels that do not provide confidentiality protection even if the confidentiality of the original data is to be protected.

4.1.7 Data decryption

A data decryption application, as well as a data encryption application using symmetric cryptography, needs access to the private (secret) key with the confidentiality and integrity protection requirements mentioned above. The output of a data decryption application may also require confidentiality protection.

4.2 Security Model

4.2.1 Objects

It can be seen in the examples in the previous section that we have to deal with data with varying requirements regarding the confidentiality and integrity protection. As far as the confidentiality is concerned, we can classify the data into three basic categories:

- public data,
- normal data – *C-normal*,
- data that are sensitive regarding their confidentiality – *C-sensitive.*

The public data require no confidentiality protection. They may be freely transmitted via communication channels and/or to remote systems that provide no confidentiality protection. An example of public data is the data downloaded from public Internet.

The normal data are to be protected by means of discretionary access control against unauthorized reading by other users than the owner of the data.

The C-sensitive data are the data that their owner (a user) wishes to remain unreadable to the others regardless of the software the user uses, and even if the users makes some mistakes (such as setting wrong access rights for discretionary access control). Examples of C-sensitive data are private and secret keys, passwords for Internet banking, etc.

As far as the integrity (or trustworthiness) of data is concerned, we can also classify the data into three basic categories:

- potentially malicious data,
- normal data – *I-normal*,
- data that are sensitive regarding their integrity – *I-sensitive.*

The requirements of the integrity protection of data is tightly coupled to the trustworthiness of the data. The trustworthiness of data can be thought of as a metric of how reliable the data are. If some data can be modified by anyone, they cannot be trusted not to contain wrong or malicious information. If some data are to be relied on, their integrity has to be protected.

The potentially malicious data require no integrity protection, and can neither be trusted to contain valid information, nor can be trusted not to contain malicious content.

The normal data is to be protected by means of discretionary access control against unauthorized modification by other users that the owner of the data.

The I-sensitive data are the data that their owner wishes to remain unmodified by the others regardless of the software the user uses, and even if the users makes some mistakes. The I-sensitive data are to be modifiable only under special conditions upon their owner's request. A special category of I-sensitive data is the category of the shared system files such as the programs, the libraries, various system-wide configuration files, the user database, ... Some of these files may be modifiable by the designated system administrator, some of them should be even more restricted.

The number of the confidentiality and integrity categories may be higher in real systems. The three levels described above have a general and easy to understand meaning, and we will use them in this section to explain the ideas of our new model.

4.2.2 Subjects

It follows from the examples in the previous section, that we have to deal with subjects with varying levels of how much we can trust them to do what they are expected to do. While we cannot trust a malicious application to do anything good, we will clearly have to trust a signature creation application to create correct signatures, for example.

A common approach to ensuring the confidentiality and/or the integrity of information in systems that deal with data classified into several confidentiality/integrity levels, is to define an information flow policy, and then to enforce the policy. In order to enforce an information flow policy, subjects are divided into two categories – trusted and untrusted. A trusted subject is a subject that is trusted to enforce the information flow policy (with exceptions) by itself; an untrusted subject is a subject that is not trusted to enforce the policy by itself, and therefore the policy has to be enforced on the subject's operations by the system.

A typical information flow policy protecting confidentiality (e.g. one based on Bell-LaPadula model[10]) states that a subject operating at a confidentiality level C_S may only read from an object with a confidentiality level C_{O_r} if $C_S \geq C_{O_r}$, and may only write to an object with a confidentiality level C_{O_w} if $C_S \leq C_{O_w}$. If a subject is to be able to read from a more confidential object, and to write to a less confidential object, it has to be a trusted subject.

A typical information flow policy protecting integrity (e.g. one based on Biba model) states that a subject operating at an integrity level I_S may only read from an object with an integrity level I_{O_r} if $I_S \leq I_{O_r}$, and may only write to an object with an integrity level I_{O_w} if $I_S \geq I_{O_w}$. Only a trusted subject can read from an object with a lower integrity level, and write to an object with a higher integrity level.

The problem with the division of subjects into the two categories is that it would lead to the need of too many trusted subjects in the home and office environment. Considering the examples given in the previous section, many of the identified applications would have to be trusted.

We will divide subjects into three categories:

- untrusted subjects,
- partially trusted subjects, and
- trusted subjects.

An untrusted subject is a subject that is not trusted to enforce the information flow policy. It is assumed to perform any operations on any objects unless it is prevented from doing so by the operating system.

A trusted subject is a subject that is trusted to enforce the information flow policy by itself. A trusted subject may be used to perform tasks than require violation of the policy under conditions that are verified by the trusted subject. A trusted subject can, therefore, be used to implement an exception to the policy.

A partially trusted subject is a subject that is trusted to enforce the information flow policy regarding a specific set of objects, but not trusted to enforce the information flow policy regarding any other objects. In other words, a trusted subject is

- trusted not to transfer information from a defined set of objects (designated inputs) at a higher confidentiality level to a defined set of objects (designated outputs) at a lower confidentiality level in a way other than the intended one, and
- trusted not to transfer information from a defined set of objects (designated inputs) at a lower integrity level to a defined set of objects (designated outputs) at a higher integrity level in a way other than the intended one, but
- not trusted not to transfer information between any other objects.

The sets of designated inputs and outputs regarding confidentiality are distinct from the sets regarding integrity. Any of the sets may be empty. A partially trusted subject, like a trusted one, can be used to implement an exception to the policy, because it can violate the policy (and it is trusted to do it only in an intended way).

The most important difference between trusted and partially trusted subjects is in the level of trust. While trusted subjects are completely trusted to behave correctly, partially trusted subjects are only trusted not to abuse the possibility of the information flow violating the policy between a defined set of input objects and a defined set of output objects.

4.2.3 Information Flow Policy

Having specified the objects and the subjects and their classification, we can formulate the information flow policy to protect the confidentiality and the integrity of the information stored in, or transferred via the objects. We will first specify the policy objectives in an informal way, and then we will define the policy formally.

In accordance with the classification of objects, the information flow policy has the following objectives:

1. Prevent reading of C-sensitive objects by subjects of other users than the owner of the object.
2. Prevent modification of I-sensitive objects by subjects of other users than the owner of the object.
3. Prevent information passing from objects with a higher confidentiality level[3] to objects with a lower confidentiality level by untrusted subjects with the exception stated below.
4. Allow the user to explicitly allow a subject to read a C-normal object on per request basis. The user's approval in such case must be obtained via a mechanism independent on the subject.

   The idea of this objective is to allow the user to perform operations such as submitting a C-normal document to a remote system, that is not trusted to process C-normal data in general and is considered a public object with respect to our classification scheme, without the need to reclassify the document first (a therefore to expose its content to any subject). Because this approach is very prone to the user's mistakes, it should be limited to C-normal objects and not applicable to C-sensitive objects.
5. Prevent information passing from objects with a lower integrity level[4] to objects with a higher integrity level by untrusted subjects.
6. Allow the user to specify the maximal integrity level for each subject and prevent the subject from writing to objects with a higher integrity level.

   The idea of this objective is to prevent modification of objects with a high integrity level unless required by the user.
7. Allow the user to define four sets of special input and output objects (two sets for confidentiality protection and two sets for integrity protection) and two special confidentiality levels (for reading and writing respectively) and two special integrity levels associated with the sets for each partially trusted subject, and apply the same rules to partially trusted subjects with the following exceptions:

   (a) Allow a partially trusted subject to transfer information from an object O_{in} with a confidentiality level c_{in} to an object O_{out} with a confidentiality level $c_{out} < c_{in}$ if the object O_{in} is in the input set for confidentiality protection, the object O_{out} is in the output set for confidentiality protection, c_{in} is at most the special

[3]Assuming the ordering of the confidentiality levels as follows: public < C-normal < C-sensitive.

[4]Assuming the ordering of the integrity levels as follows: potentially malicious < I-normal < I-sensitive.

confidentiality level for reading, and c_{out} is at least the special confidentiality level for writing.

(b) Allow a partially trusted subject to transfer information from an object O_{in} with an integrity level i_{in} to an object O_{out} with an integrity level $i_{out} > i_{in}$ if the object O_{in} is in the input set for integrity protection, the object O_{out} is in the output set for integrity protection, i_{in} is at least the special integrity level for reading, and i_{out} is at most the special integrity level for writing.

8. Allow the user to define the maximal confidentiality level for reading C_S^{max}, the minimal confidentiality level for writing C_S^{min}, the minimal integrity level for reading I_S^{min} and the maximal integrity level for writing I_S^{max} for each trusted subject S, and allow the trusted subject S to read from an object O with a confidentiality level C_O and an integrity level I_O only if $C_O \leq C_S^{max} \wedge I_O \geq I_S^{min}$, and allow the trusted subject S to write to the object O only if $C_O \geq C_S^{min} \wedge I_O \leq I_S^{max}$.

   The trusted subjects are, therefore, able to transfer information between any objects within some limits.

In the thesis, we present a security formal model described by a logical function for each operation. The functions operate on subjects and objects and determine whether the particular operation is allowed. Every object is assigned 4 attributes, every subject is assigned 18 attributes, and these attributes are used to specify the logical functions.

In the thesis, we also present formally specified and proved theorems stating the security properties of our model[5]. The model prevents untrusted subjects from causing any information flow from an object with a higher confidentiality / lower integrity level to an object with a lower confidentiality / higher integrity level. It also ensures that partially trusted subjects are able to cause such information flow only if they obey the rules mentioned above.

4.3 Protection Profile Overview

To fulfil one of the goals of the thesis, we have created a protection profile, compliant to the ISO/IEC 15408 standard[3, 4, 5], for a general purpose operating system suitable for using security-critical applications alongside potentially malicious ones.

Our protection profile has been designed to support our new security model in addition to the common discretionary access control policy supported by the common existing operating systems.

Our protection profile specifies several assumptions about the security environment of the operating system, namely about the hardware and its physical surroundings:

- the hardware supports access control for memory regions and peripheral devices,
- the processor(s) restrict the use of privileged instructions to the operating system,
- the hardware is physically protected against modification,
- and the internal communication paths (such as system buses) are protected against unauthorized monitoring and tampering with.

The hardware assumptions are consistent with the currently common PC hardware. The other two assumptions are to be upheld by the environment, most commonly using physical security mechanisms.

It is also assumed that an operating system compliant with the protection profile is constructed as a modification of an existing operating system. On one hand this limits the choice of security functional requirements and, most notably, security assurance requirements; on the other hand it allows a compliant operating system to be used in practice by utilizing existing applications and hardware support.

The objectives of the protection profile can be summarized as follows:

- restricting access to identified, authenticated, and authorized users only,
- enforcing a discretionary access control policy based on user identities,
- enforcing confidentiality and integrity protection according to our new security model,
- cryptographic confidentiality and integrity protection of stored data,
- auditing of security related events,
- residual information protection,
- system management restricted to authorized system administrators,
- limiting the ability of an authorized administrator to abuse his/her privileges,
- and the feasibility of construction of a compliant operating system by modifying an existing operating system while preserving its compatibility with existing applications.

The security functional requirements specified in the protection profile can be summarized as follows:

- Security Audit (FAU) – the protection profile requires that the operating system is able to generate audit records for the listed events, that the authorized audit administrators are able to review, sort and search the audit records, and configure the set of audited events, and that the audit records are protected against unauthorized modification and deletion.
- User Data Protection (FDP) – the protection profile requires that the operating system enforces the discretionary access control policy and our information flow policy based on the user identity and group membership and other security attributes associated with subjects and objects. It also requires

[5]The theorems can also be found in [14].

removal of any residual information from resources upon their allocation. Support for cryptographic protection of integrity and confidentiality of stored data is also required.

- Identification and Authentication (FIA) – the protection profile specifies security attributes to be associated with each user and the user's processes. It allows minimal strength requirements for authentication data to be specified. It requires a successful identification and authentication of an authorized user before performing any other action on the user's behalf. It requires re-authentication before performing several specific security-critical actions.
- Security Managements (FMT) – the protection profile requires the management of the security attributes and data to be restricted to the authorized users acting in specific roles, and in compliance with the information flow and access control policies. The protection profiles defines several security roles and splits the privileges and responsibilities among them in order to minimize the ability of a single individual (e.g. a system administrator) to abuse his/her privileges. It also supports authorization of security-critical operations by multiple administrators.
- Protection of TOE Security Functions (FPT) – the protection profile requires the operating system to protect itself against tampering by unauthorized subjects, as well as to separate the security domains of subjects. The protection profile requires that the security functions must always be invoked before other operations within the scope of control can be performed, i.e. it may not be possible to bypass the security functions. The protection profile also requires the operating system to perform tests during start-up to verify the crucial assumptions about the hardware.
- Resource Utilization (FRU) – the protection profile requires the operating system to enforce the maximum quotas on disk space used by an individual user.
- TOE Access (FTA) – the protection profile requires the operating system to support session locking on the user's request as well as automatically after a specified period of inactivity. It also requires the operating system to maintain and display information about successful and unsuccessful session establishments.
- Trusted Path/Channel (FTP) – the protection profile requires the operating system to provide a trusted path between the user and itself for specified security-critical interactions including (re-)authentication, authentication data management, special security attributes management, ...
- Cryptographic Support (FCS) – the protection profile specifies requirements for cryptographic key generation and destructions, and for cryptographic operations.

The security assurance requirements specified in the protection profile are based on EAL3[5] augmented with the addition of requirements for detection of modification during delivery of the operating system.

5. Discussion

In the thesis, we discuss the benefits of our model. We present several usage examples based on the considered classes of applications. These examples can also be found in [13]. We also compare our new security model and our protection profile to several other projects and to the mentioned existing protection profiles.

5.1 Our New Model vs. MIC in Windows Vista

MIC can be used to prevent a potentially malicious application running at a low integrity level from modifying data at a higher integrity level. Unlike our new model, however, it does not prevent an untrusted application running at a higher integrity level from reading (potentially malicious) data at a lower level. If the application contains a flaw that can be exploited by processing malicious data, the user can, e.g. by an accident, use it to read and process malicious data stored at the low integrity level, and thus turn the application to a malicious one that can spoil data at its (higher) integrity level.

MIC also allows confidentiality protection to be turned on. Unlike in our new model, the confidentiality and integrity levels in MIC are not independent, the same level is used for both. It can be used to prevent a potentially malicious application running at a low level from reading (and also from modifying) data at a higher level. Unlike our new model, it does not prevent an application running at a higher level (and thus capable of reading data at that level) from writing to objects at a lower level. Any application is thus able to export any data that it can read to external untrusted systems, or to store them to a low-level file.

MIC is a useful feature that allows a careful user to use a web browser or to test a potentially malicious application without the risk that they will modify (and optionally also read) any data classified at a higher level. The user must be careful enough, however, not to open any files classified at the low level in another application running at a higher level unless the application is trusted not to misbehave upon reading the data.

The integrity and confidentiality protection provided by our new security model is definitely stronger than that of MIC, and it has provable security properties similar to those of Bell-LaPadula and Biba models.

5.2 Our New Model vs. SELinux

SELinux, by implementing a flavour of domain and type enforcement, provides a very flexible security mechanism that can be used to enforce a wide range of security policies. We show in the thesis that it can be used to implement our new model as well.

The commonly used SELinux policies[11, 2] define unique types and domains for many typical system services, server applications, and their data, and strictly restrict the set of operations the processes running within these domains are allowed to perform. The strict version of the policy is suitable for servers but causes problems on desktop systems because the restrictions are too strict to be accepted by users. The targeted version of the policy, that restricts many system services but leaves the applications started by the user in a single, unconfined domain, is suitable for desktops. It prevents a flawed system service program from accessing data it does not have to be able to access.

It does not, however, prevent user-started applications from accessing the user's data, perhaps except for some specially designated sensitive data (such as private keys) accessible only to certain applications.

The targeted policy could be combined with our new model to provide combined benefits of both. The targeted policy provides more rigorous restrictions for specific services while our new model can be used to protect the ordinary users' data.

5.3 Our Protection Profile vs. CAPP and LSPP

Our protection profile, the CAPP, and the LSPP are designed for general purpose operating systems. We will now discuss the benefits of our protection profile when compared to those two.

All three protection profiles have some common objectives, such as restricting access to authorized users only, audit, discretionary access control policy enforcement, residual information protection and restricting system management to authorized administrators only. They also use similar security functional requirements to fulfil these objectives. We will, therefore, concentrate on the objectives that make the protection profiles different.

There are three new, important objectives in our protection profile when compared to the CAPP or the LSPP:

- enforcing confidentiality and integrity protection according to our new security model,
- cryptographic confidentiality and integrity protection of stored data,
- and limiting the ability of an authorized administrator to abuse his/her privileges.

The first is intended to address the problem (discussed also in the section 2.1) of abusing the privileges of a user (including an administrator) by malicious code (whether executing as a standalone malicious applications, or as a part of a flawed application). Our new model has provable security properties that ensure that no untrusted applications can cause information to flow from objects at a higher confidentiality and/or lower integrity level to objects at a lower confidentiality and/or higher integrity level. The user can, thus, run a malicious application as an untrusted subject without the risk that it could cause an information leak or spoiling. Applications that need to be able to override this restriction can often be run as partially trusted subjects where the amount of trust can be very limited. We expect this approach to minimize the risk resulting from flaws in such applications.

The second new objective addresses the problem of direct manipulation with storage devices. Data encryption provides for confidentiality protection of the stored data against manipulation by means that are not under control of the operating system. Cryptographic integrity protection can prevent unnoticed unauthorized modification of the stored data including the operating system itself.

Our protection profile is missing the assumption of a trustworthy administrator (this assumption is present in both the CAPP and the LSPP). The third objective addresses the problem of privilege abuse by an administrator. This is not addressed by the other two protection profiles. In our protection profile, it is addressed by separating the privileges and responsibilities for various aspects of system management (such as general system management, security management, audit management), and by defining distinct security roles for such activities. When the roles are assigned to different individuals, no single person can cause an unnoticed security policy violation. For environments, where this problem is a major concern, our protection profile supports multiple independent authorizations for critical operations.

When compared to the LSPP's Bell-LaPadula style confidentiality protection, our protection profile addresses the integrity protection as well as the confidentiality protection. The integrity protection is often at least as important as the confidentiality protection, as we have argued in the section 4.2. The concept of the partially trusted subjects, that we have introduced, is also a convenient way of minimizing the amount of trust given to applications that have to be able to violate the rules for untrusted subjects.

5.4 Benefits Summary

The key benefits of our new security model are summarized in the table 1. While we have based our new model on the ideas of Bell-LaPadula and Biba models[10, 20], we have introduced significant improvements. Unlike others (e.g. MIC[18]), we use independent confidentiality and integrity levels, and, most notably, we have introduced partially trusted subjects. The introduction of partially trusted subjects allows us to achieve strong security properties with minimal trust in correct behaviour of the partially trusted subjects.

Our protection profile, unlike the well-known and often used protection profiles (CAPP[16] and LSPP[17]), addresses the issues of privilege abuse by a system administrator, includes cryptographic protection against direct storage media manipulation, and provides for both confidentiality and integrity protection according to our new security model.

5.5 Feasibility of Implementation

The thesis includes a feasibility study to show that it is feasible to modify a real, existing operating system to comply with our protection profile. We have chosen Linux operating systems as the base. We compare the security functions of Linux operating systems with the security functional requirements of our protection profile, and identify the missing features.

We suggest two ways of implementing our new security model – the major missing feature – in Linux. In one approach we suggest using Linux Security Module framework present in the Linux kernel. In the other one we show how a SELinux policy can be created to enforce the rules of our new security model (see also [15].

6. Conclusions

The goals of our thesis were to design a suitable security model and to create a protection profile for a general purpose operating system for home and office environment suitable to support the use of security-critical applications alongside untrusted and potentially malicious applications. We have presented several typical examples

	our new model	Bell-LaPadula	Biba	MIC
confidentiality protection	✓	✓		✓
integrity protection	✓		✓	✓
prevents untrusted subjects from causing information leak/spoiling	✓	✓	✓	
prevents partially trusted subjects from causing arbitrary information leak/spoiling	✓			

Table 1: Summary of the benefits of our new security model

of applications used in the target environment, analyzed the security requirements and properties of the data involved, and designed a security model with a formally defined information flow policy to protect the confidentiality and the integrity of the data. In the thesis, we have formally proved important security properties of the model. We have created a new protection profile utilizing our new security model. We have discussed the benefits of our new security model and protection profile for security. We have compared them to other projects and protection profiles. Finally, we have discussed possible ways to modify Linux operating system to comply with our protection profile. It seems to be feasible to make the needed modifications in a near future. Having said that, we believe we have managed to fulfil the goals.

We can see the following tasks that might follow the thesis:

- Implementation – attempt to implement the suggested modifications. This is currently a work in progress of two of our students, and we expect to have the first results in June, 2010.
- Usability testing – find out whether the resulting system can be used without unacceptable discomfort for the users.
- Compare the different possible ways to make the needed modifications in terms of performance, usability, compatibility, extensibility, ...

Acknowledgements. This contribution has been supported by the grant VEGA 1/0266/09.

References

[1] Domain and type enforcement. http://www.cs.wm.edu/~hallyn/dte/. [cit. 2004].

[2] Fedora selinux project : discussion of policies. http://fedoraproject.org/wiki/SELinux/Policies. [cit. 2009].

[3] ISO/IEC 15408:2005, common criteria for information technology security evaluation : part 1 introduction and general model. http://www.commoncriteriaportal.org/.

[4] ISO/IEC 15408:2005, common criteria for information technology security evaluation : part 2 security functional requirements. http://www.commoncriteriaportal.org/.

[5] ISO/IEC 15408:2005, common criteria for information technology security evaluation : part 3 security assurance requirements. http://www.commoncriteriaportal.org/.

[6] Linux intrusion detection system. http://www.lids.org/. [cit. 2004].

[7] Medusa DS9. http://medusa.fornax.sk/. [cit. 2004].

[8] Role set based access control. http://www.rsbac.org/. [cit. 2004].

[9] Security enhanced linux. http://www.nsa.gov/selinux/. [cit. 2004].

[10] D. E. Bell and L. J. La Padula. Secure computer system : unified exposition and multics interpretation : technical report. http://csrc.nist.gov/publications/history/bell76.pdf, 1976.

[11] F. Coker and R. Coker. Taking advantage of selinux in red hat enterprise linux. *Red Hat Magazine*, (6), 2005. [cit. 2009].

[12] J. Janáček. Bezpečnosť operačných systémov : písomná časť dizertačnej skúšky. http://www.dcs.fmph.uniba.sk/~janacek/BezpecnostOS.pdf.

[13] J. Janáček. Mandatory access control for small office and home environment. In *Informačné Technológie – Aplikácie a Teória : Zborník príspevkov prezentovaných na pracovnom seminári ITAT*, pages 27–34, Seňa, 2009. PONT s.r.o.

[14] J. Janáček. A security model for an operating system for security-critical applications in small office and home environment. *Communications : Scientific Letters of the University of Žilina*, 11(3):5–10, 2009. ISSN 1335-4205.

[15] J. Janáček. Two dimensional labelled security model with partially trusted subjects and its enforcement using selinux dte mechanism. In *Networked Digital Technologies, Part I*, volume 87 of *Communications in Computer and Information Science*. Springer, 2010. ISBN 978-3-642-14291-8, to appear.

[16] NSA. Information systems security organization: Controlled access protection profile : version 1.d. http://www.commoncriteriaportal.org/files/ppfiles/capp.pdf, 1999.

[17] NSA. Information systems security organization: Labeled security protection profile : version 1.b. http://www.commoncriteriaportal.org/files/ppfiles/lspp.pdf, 1999.

[18] S. Riley. Mandatory integrity control in windows vista. http://blogs.technet.com/steriley/archive/2006/07/21/442870.aspx. [cit. 2009].

[19] M. Russinovich. Inside windows vista user account control. http://technet.microsoft.com/en-us/magazine/2007.06.uac.aspx. [cit. 2009].

[20] H. F. Tipton and M. Krause, editors. *Information Security Management Handbook*. CRC Press LLC, 5th edition, 2004. ISBN 0-8493-1997-8.

Selected Papers by the Author

J. Janáček, R. Ostertág. Problems in Practical Use of Electronic Signatures. In S. Fischer-Hübner, D. Olejar, K. Rannenberg (eds.), *Security & Control of IT in Society - II – Proceedings of the IFIP WG 9.6/11.7 Working Conference*, pages 63–74. Bratislava, IFIP, 2001.

J. Janáček, R. Ostertág. Riziká Internet bankingu. In *6. konferencia s medzinárodnou účasťou Informatika 2001 – zborník prednášok*, pages 145–150. Bratislava, 2001.

J. Janáček. Security Aspects of Electronic Signatures. In H. D. Zimmermann (ed.), *Proceedings of SOFSEM 2001 Workshop on Electronic Commerce Systems*, pages 15–20. Bratislava, Faculty of Mathematics, Physics and Informatics, Comenius University, 2001.

J. Janáček, R. Ostertág. Problems in practical use of electronic signatures. In *Informatica*, 2002, vol. 26, no. 2, pages 151–157.

J. Janáček. Výhody a riziká používania elektronického podpisu. In *Moderné informačné systémy 2003 – zborník príspevkov*. Bratislava, Softec, 2003.

J. Janáček. A Security Model for an Operating System for Security-Critical Applications in Small Office and Home Environment. In *Communications – Scientific Letters of the University of Žilina*. 2009, vol. 11, no. 3, pages 5–10.

J. Janáček. Mandatory Access Control for Small Office and Home Environment. In P. Vojtáš (ed.), *Informačné Technológie – Aplikácie a Teória – Zborník príspevkov prezentovaných na pracovnom seminári ITAT*, pages 27–34. Seňa, PONT s.r.o., 2009.

J. Janáček. Two dimensional labelled security model with partially trusted subjects and its enforcement using selinux dte mechanism. In *Networked Digital Technologies, Part I*, volume 87 of *Communications in Computer and Information Science*. Springer, 2010. ISBN 978-3-642-14291-8, to appear.