Security architectures for network clients

Victoria Skoularidou
Systems Engineer, Department of Management Science and Technology, Athens University of Economics and Business (AUEB), Athens, Greece
Diomidis Spinellis
Assistant Professor, Department of Management Science and Technology, Athens University of Economics and Business (AUEB), Athens, Greece

Keywords
Security, Computer architectures, Networks, Clients, Software

Abstract
Enumerates and compares a number of security-enabling architectures for network clients. These architectures, either proposed as methodologies or currently implemented in software and/or hardware, are capable of protecting the client's software integrity and its environment. The most important methodologies include the reference monitor model, firewalls, and virtual machines. Software implementations are the Java Sandbox, and the code signing concept. Hardware that can be used includes smart cards, and the TCPA/Palladium security initiative. Describes their most important features and provide a review and comparative study based on a number of criteria. Believes that ongoing research can empower these mechanisms for protecting network clients in a more effective way.

An earlier version of this work appeared in the Proceedings of the 3rd International Network Conference (INC 2002), 16-18 July, Plymouth, UK, pp. 389-96.

Introduction

Despite effort being expended to secure network clients, these are increasingly and continuously succumbing to malicious software (viruses, worms, and Trojan horses). As the same client is nowadays trusted to conduct financial transactions or store and process sensitive personal information, users deserve to be assured of a higher level of security than what is currently the norm. In this paper we review, from an architectural standpoint, methodologies and technologies that can be used towards this end.

According to Ghosh (1998) the security of Web-based systems should be ensured in four fronts: Web client, data transport, Web server and operating system security. In this survey paper, we focus on the network client side and examine a number of architectures and technologies that can be used for protecting the integrity of the clients and their environment. With the term "client" we refer to Web clients, e-mail clients, access clients (like ftp and/or telnet), and similar applications. These architectures have either been proposed as methodologies, presented in the next section, or are actual implementations (in software and hardware) currently in use and described in the following two sections. The final section draws the lessons of this study and compares the security-enabling architectures that were studied.

Methodologies

From the theoretical security models in existence, some have been realized in commercial product implementations while others were abandoned and exist only as concepts in the research community. In the first category we can identify the notion of the firewall and virtual machine while the reference monitor model falls in the second one. We provide a description of these models in the following paragraphs.

The reference monitor security model

The reference monitor was based on the abstract modeling efforts of Lampson (1971) and was also described by Anderson (1972). It is depicted in Figure 1 (Stallings, 1995).

The reference monitor is a controlling element in the hardware and operating system of a computer that regulates the access of subjects (e.g. users, processes, etc.) to objects (e.g. files, programs, etc.), on the basis of their security parameters. It has access to the security kernel database (SKDB) that lists the access privileges of each subject and the protection attributes of each object. Any detected security violations and authorised changes, are stored in an audit file.

The reference monitor concept has inspired research in the area of inline reference monitors and language-based security (Erlingsson and Schneider, 1999, 2000).

One major problem of the reference monitor concept is that it is too complex and requires the developer to start with a totally new operating system (and probably hardware) design (Lobel, 1986). Another problem is that early attempts to reproduce it in actual hardware and software met with only minimal success, primarily because of unexpectedly high overhead and/or system performance degradation. One historical example is the operating system MULTICS (Organick, 1972), developed in the late 1960s by MIT, Bell Labs, and Honeywell.

However, supposing that the reference monitor was implemented as a part of a system, then a network client could be protected in the following way: let's imagine

Information Management & Computer Security
11/2 [2003] 84-91
© MCB UP Limited
[ISSN 0968-5227]
[DOI 10.1108/09685220310468664]

that a UNIX system user navigates with the Web browser into a number of Web sites. According to the reference monitor's policy, described in the SKDB, as a subject, the only privilege the user has is the capability of saving Web pages, files, etc. in the directory "Internet_files" of the mounted hard disk (the corresponding object). If a malicious applet is downloaded on the user's machine and tries to gain root privileges by, e.g. executing a SUID program, it will simply fail since the reference monitor will deny access, according to the previous security policy. The same applies with the user's mail client. If the user is only allowed to save attachments on the disk storage then a rogue program could not harm the system, as the reference monitor would prevent any compromise.

The firewall concept

Properly configured firewalls can constitute an effective type of network security. They prevent the dangers of the Internet from spreading into the internal network by restricting access at a centrally managed point.

Firewalls are classified into three main categories (Cheswick and Bellovin, 1994):

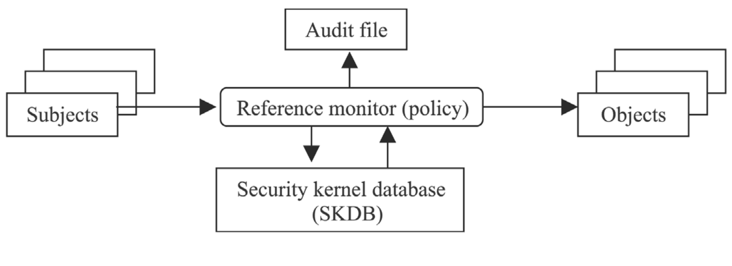
1 packet filters that drop packets based on their destination address and port;
2 circuit gateways that relay TCP connections; and
3 application-level gateways where special-purpose code is used for each desired application (making it easy to log and control all incoming and outgoing traffic).

Application-level gateways can provide a centralized point for monitoring the behavior of an electronic mail system and they can analyze and record traffic and content looking for information leaks. Their principal disadvantage is the need for a specialized user program for most services provided. Also, the use of such gateways is easiest with applications that make provision for redirection, such as e-mail, otherwise new client programs must be provided.

Another category of firewalls, becoming increasingly popular, is the personal firewalls that can be useful for preventing and even detecting potential spyware and also protecting from malicious executables (Ghosh, 2001). However, they cannot help in the detection of spyware that is masquerading in programs that use the network for other legitimate purposes.

To recapitulate, firewalls are not panacea, since they must be properly configured and regularly updated as new threats and vulnerabilities are discovered (Zwicky *et al.*, 2000). They offer only one layer of protection and cannot be considered a full security solution since they cannot protect from insider attacks (Garfinkel and Spafford, 2002) and cannot block encrypted information or traffic tunneled via HTTP although some solutions have been provided (Martin *et al.*, 1997).

Figure 1
The reference monitor concept



The virtual machine concept

A virtual machine is a piece of computer software designed to reproduce a specific set of computer behaviors and capabilities other than the ones native to the computer or operating system on which the software itself is running. Some virtual machines are emulators; others produce behaviors and capabilities of a machine that doesn't necessarily exist as an actual piece of hardware but may only be a detailed specification. More modern examples include the specification of the Java Virtual Machine (JVM) (Lindhorn and Yellin, 1997) and the common language infrastructure of the Microsoft.NET initiative. These allow diverse computers to run software written to that specification; the virtual machine software itself must be written separately for each type of computer on which it runs. Other virtual machines let one operating system run on top of another on the same machine (VMware Inc., 2000).

The virtual machine design has two advantages:

1 system independence, since any application will run the same in any virtual machine, regardless of the hardware and software underlying the system;
2 security, because the virtual machine has no contact with the operating system, hence there is little possibility of a program damaging other files or applications.

The virtual machine can be used to sandbox applications since it stands between the real hardware or another operating system layer (the virtual machine is often an operating system). This, of course, has a downside concerning efficiency, because operating system calls and privileged instructions of programs running in a virtual machine have to pass through the virtual machine layer. Thus, virtual machines like JVM and VMware also provide a restricted environment in which programs may operate. Errant applications should only be able to cause damage to the virtual machine, thus leaving the real system intact.

Software implementations

Here, we present network client security architectures currently implemented in software that allow the secure execution of downloadable executable content (i.e. mobile code).

The Java Sandbox

The concept of sandbox or software fault isolation was first described in (Wahbe *et al.*, 1993). The Java Sandbox is Java's security model, by which any untrusted Java applet must abide. It is a technological solution to prevent malicious code behavior, thus protecting a network client from possible attacks. For example, if a user downloads via the Web browser an applet that tries to erase the user's hard disk, it will fail because the sandbox restricts its operation, since it is untrusted. The Java Sandbox is enforced by three technologies:

1 the bytecode verifier;
2 the applet class loader; and
3 the security manager (McGraw and Felten, 2000).

The Java Sandbox is quite complicated but it is one of the most complete existing security models. The problem is that the three technologies comprising the model work in concert to prevent an applet from abusing its restricted privileges. They are highly interdependent and non-overlapping. Because each one provides a different function, a flaw in one can break the whole sandbox (McGraw and Felten, 1997). So, their design must be solid, and their implementations must not be flawed. The complexity of the functions that each technology provides makes a correct implementation a difficult goal to attain. The Java security problems found to date are a direct result of flaws in these functions' implementations (Ghosh, 1998).

The Java security model continued to evolve with new Java releases (Gong *et al.*, 1997). JDK 1.2 introduced a more flexible security model in which the class loader can assign a different security policy to each class as it is loaded and stack inspection (Wallach and Felten, 1998) is used to determine what privileges are enabled. It also introduced the notion of protection domains and the access controller as a more abstract and flexible alternative to the security manager.

Code signing

Modern component-based software is a lot harder to secure because:

1 one cannot assume that all the modules are trustworthy;
2 one cannot assume that all the modules are written well enough to work in every possible configuration; and
3 the operating system is not there to deal with 1 and 2, since modern components talk to each other directly, not through the operating system, so any built-in safety features simply do not apply.

Several general methods for dealing with this security problem have been tried, like code signing. The programmer signs components and the user decides, based on the signatures, which components to allow on the computer. Sun's Java and Microsoft's ActiveX controls provide code-signing features.

The Java Sandbox very simply and strictly prevents Java applets downloaded from the network from using sensitive system services. The security policy for untrusted applets is black-and-white (Ghosh, 1998): if applets are downloaded across a network connection, they must abide by the strict constraints of the sandbox; if they are loaded from the local file system, they are completely trusted and given free rein of the system, as Java applications do.

To provide greater flexibility to run Java applets in a trusted environment, JavaSoft has provided the ability to sign applets using JDK's 1.1 Crypto API. It provides the ability to digitally sign applets with unforgeable proof of identity (Gritzalis *et al.*, 1998). In this way, applets access system resources based on who signs them. The black-and-white security policy for executing applets in JDK 1.1 changed to a shades-of-gray model in JDK 1.2 where more fine-grained access control is supported.

ActiveX is a framework for Microsoft's software component technology that allows programs encapsulated in units called controls to be embedded in Web pages (Ghosh, 1998). Unlike Java, ActiveX is language independent but platform specific. The controls can be written in several different languages but can be executed only on a 32-bit Windows platform. Since ActiveX controls have the ability to execute much like any other program on a computer, they may be used to forge e-mail and write files (integrity loss), monitor Web usage, send files over the Internet and interact with other programs (threat to privacy and confidentiality through information leaks), etc.

Microsoft's response to addressing ActiveX technology security problems is Authenticode (Microsoft Corp., 2001). This does not prevent ActiveX controls from behaving maliciously but it can be used to prevent automatic execution of untrusted ones. Authenticode can provide two checks before executing ActiveX controls: it can verify who signs the code (authentication), and if the code has been altered since it was signed (integrity). Authenticode provides verification of the identity of the person who signed the control and integrity checks of the software to ensure it has not been altered since it was signed. However, the signature provides no assurance that the control will not behave maliciously. Authenticode technology works solely on a trust model and there is no middle ground to let the control execute in a constrained environment where it can be observed before granting full access.

The key difference in security between ActiveX controls and Java applets is that ActiveX security is based wholly on the trust placed in the code signer, while Java applet security is based on restricting the behavior of the applet (Ghosh, 1998). One is a human judgment-based approach to security, while the other is a technology-based approach using the sandbox solution. Java applets signing has been also introduced by JavaSoft as a policy based on trust and human judgment. Signed applets have the ability to access system resources based on who signed them, but untrusted ones can still execute, albeit with the sandbox limitations.

Other techniques for trying to provide proofs in software code include proof carrying code (Necula and Lee, 1996) and efficient code certification described in Kozen (1998).

In summary, code signing does prove the integrity and authenticity of a piece of software purchased in a computer store or downloaded over the Internet. But it does not promote accountability, because it is nearly impossible to tell if a piece of software is malicious or will behave in a malicious manner (Garfinkel and Spafford, 2002). Research in certifying software components for security properties has been conducted (Ghosh and McGraw, 1998).

Hardware implementations

So far, security-enabling architectures that were proposed as methodologies or are based on software implementations were examined. In this section, we describe hardware-based ones.

Smart cards

A smart card stores and processes information through the electronic circuits embedded in silicon in the plastic substrate of its body. There are two basic kinds of smart cards (Chen, 1998): an intelligent smart card contains a microprocessor and a memory chip and offers read, write, and calculation capability. A memory card contains only a memory chip, is meant only for information storage and can only undertake a predefined operation. Smart cards can carry all necessary functions and information on the card, so they do not require access to remote databases at the time of the transaction.

Their benefits of increased storage, security and portability have made them very popular compared with magnetic stripe cards, that are not so secure, require a host system to store and process all data and cannot make data universally accessible (Coleman, 1998). By putting sensitive information like passwords and encryption keys into a central point like the card and, thus, outside of the client's environment, the client becomes less vulnerable to malicious attacks. Typically any application requiring authentication can benefit from a smart card. Smart cards can be used for authentication and as a secure, convenient portable storage mechanism.

On the other hand there also exist problems: if a hacker takes the control of the client he could force the card to do something the client does not want like giving his credit card information to a malicious site (Balfanz and Felten, 1999). Also, since smart cards blindly sign any data that is sent to them the user has no way of verifying that this data is what he wanted to be signed (Freudenthal *et al.*, 2000). In such a case a hacker could modify the signing software so that it makes changes to a document before it is signed. As a result the user may see one document, but sign something else.

With the advent of the Java Card (a smart card capable of running Java bytecodes) limitations like the portability of applications and the flexibility of downloading applications into the card are eliminated, since a single Java application can run on all smart cards (Coleman, 1998). Since one of the fundamental problems in securing computer systems is the need for tamper-resistant storage of keys, smart cards can provide this functionality so that the private key of the network client can be placed on it and the access control on the card is offered via a proper personal identification number (PIN). Smart cards provide also the ability to upgrade security solutions when they become compromised, e.g. if a hacker cracks the security of smart-card enabled digital satellite systems new cardlets (Java Card applications) could be sent.

The fact that smart cards now employ public key encryption to both encrypt data and digitally sign messages to provide unforgeable proof of identity, makes them ideal for integrating into them applications like social security card, access control to Web sites or online databases, digital signatures for e-mail and Web transactions, public keys for encrypting data transactions, credit/debit cards, e-cash, etc. (Ghosh, 1998). Smart cards' importance has been identified by major credit card organizations like Visa, which has recently announced its chip migration plan (Visa International, 2001) involving the substitution of credit cards with new ones with a microchip, more suitable for e-banking and e-commerce applications.

Trusted hardware

Palladium, which like the chemists, Microsoft calls "Pd" in short (Microsoft Corp., 2002), is Microsoft's implementation of the Trusted Computing Platform Alliance (TCPA) specification. The TCPA is an industry-working group, initially formed by Compaq, HP, IBM, Intel, and Microsoft in October 1999 with the mission to: "... create a new computing platform for the next century that will provide for improved trust in the PC platform", thus build a trusted computer (TCPA, 2000). TCPA now lists about 200 corporate members and has already published the TCPA Specification, v1.1.

Palladium is distinct from TCPA and does not follow the specification exactly. The idea is that a trusted computer can be built where different users on the system have limitations in their abilities and are isolated from each other (compartmentalization). This is impossible to achieve using only software, and Palladium is a combination of hardware and software modules (Schneier, 2002). Palladium and TCPA have some architectural points in common, such as the use of "trusted hardware" within a PC in order to establish a root of trust. They both require modifications to existing hardware architecture in order to work and also modifications to software in order to use trust features. As the two initiatives appear to be interrelated, in the rest of the paper we will use the term "TCPA/Palladium".

TCPA/Palladium requires changes to four parts of the PC hardware:

1 the CPU;
2 the chipset (on the motherboard);
3 the input devices (i.e. mouse, keyboard, etc.); and
4 the video output devices (graphics processor).

Additionally, a new component must be added, a tamper-resistant secure cryptographic co-processor, which Microsoft calls SCP or SPP (Schoen, 2002).

TCPA/Palladium provides protection against two broad classes of attacks:

1 remote network-mounted attacks (buffer overflows, other programming flaws, malicious mobile code, etc.); and
2 local software-based attacks (e.g. a debugger trying to read a program's internal state while executing or trying to subvert its policy).

Although TCPA/Palladium is a promising effort for providing trusted computing platforms it is not without problems. Threats to privacy, interference with GNU Public License, restriction of fair use rights (ability to copy and use copyrighted material for personal use) and the "First sales doctrine" (ability to resell software or a Palladium-equipped computer) have been extensively discussed in the literature (Anderson, 2002; Arbaugh, 2002).

Review and comparison

After presenting the various types of security-enabling architectures, in this

Table I
Protection against security threats

	Leakage	Tampering	Resource stealing	Repudiation	Malware	User ignorance
Reference monitor	✓	✓	✓		✓	✓
Firewalls	✓	✓	✓		✓	✓
Virtual machines	✓	✓	✓		✓	✓
Java Sandbox	✓	✓	✓		✓	✓
Code signing				✓		
Smart cards	✓	✓	✓		✓	✓
TCPA/Palladium	✓	✓	✓	✓	✓	✓

Table II
Non-functional characteristics of the described technologies

	Level of protection and security service provided	**Complexity**	**Ease of use**	**Incorporation into existing applications**
Reference monitor	Offers a high level of protection by residing at the lowest system layer. Adding security to the lowest level automatically secures all the above layers (Saltzer *et al.*, 1984)	Very complex since it needs new operating system design	A new operating system with system calls based on the reference monitor would be difficult to use	Presumes a new operating system (and maybe hardware) design
Firewalls	Best solution for separating the internal network but cannot provide protection against malicious insiders. An application-level gateway can provide better protection than a packet filter since it does not rely only on addresses and ports	Their installation requires the configuration of a number of devices	They need installation and configuration procedures	Their configuration can be easily provided
Virtual machines	They provide separation and isolation of processes	Realization requires the installation of a proper package	They need installation and configuration procedures	Can be easily installed on a system in order to make it capable of accessing another one
Java Sandbox	Ideal for mobile code since it can protect the integrity of the client environment by confining the use of resources	Its complexity lies in the strong interdependence of its three basic components	It needs only knowledge of the proper packages	It is ready for operation whenever mobile code (Java applet) needs to be executed on a client machine
Code signing	Ideal for mobile code since it can protect the integrity of the client environment by providing proof of origin and alteration attempt	A signature that accompanies the component is needed	It needs only knowledge of the proper packages	The only thing needed is a proper toolkit for being able to sign the code produced
Smart cards	Perfect for authentication provision	Complexity lies in the familiarization with the accompanying features (reader, use of a PIN, etc.)	The user uses them as a black box and the programmer creates the proper application	In order to operate a proper reader needs to be used and the smart card to be programmed
TCPA/Palladium	Intended to provide data security, integrity, authenticity, and privacy	Complex enough, as a number of co-operating software and hardware modules comprise the whole architecture	Promises to offer transparency to the end user	Requires hardware and software modifications in order to provide trust features

section we review and use them as a basis for a comparative study.

First, we identify the protection these mechanisms offer against specific security threats (threat model). Generally, security threats to computer systems fall into the following broad classes (Gritzalis and Spinellis, 1997; Meyer *et al.*, 1995):

- *Leakage (disclosure).* The acquisition of information by unauthorized recipients (loss of confidentiality or privacy).
- *Tampering (modification).* The unauthorized alteration of information (loss of integrity).
- *Resource stealing.* The unauthorized use of system facilities.
- *Repudiation.* Loss of attribution.

Table I summarizes the protection against these security threats offered by the described technologies. Malware and user ignorance have been added, since they also comprise serious threats to a computer system:

Apart from the level of protection and the security services that these mechanisms provide, we also compare them against a number of non-functional characteristics (Sommerville, 2001), summarized in Table II:

- *Complexity.* It is not enough to just allege that a certain methodology provides security. On the contrary, security attributes need to be easily verified thus should not be complex. Simplicity is a fundamental hint of computer systems design (Lampson, 1983).
- *Ease of use.* This is another important attribute, since usually system administrators and users do not want to use awkward systems.
- *Incorporation into existing applications.* How easily these mechanisms could be ported into existing systems.

These technologies can be combined in order to provide more fine-grained protection, based on the security services offered in different layers (at the operating system level via the virtual machine and the reference monitor, at the network level via firewalls, at the application-level via the Java Sandbox and code signing techniques, etc.) (Saltzer, 1984). For example, in the case of a Java Card application, the Java Sandbox and/or the code signing mechanism need to operate in order to prevent a malicious one from being downloaded to a smart card. Similarly, if a firewall lets applets to be executed on the client's machine, the Java Sandbox and/or code signing features should be also used to prevent a possible malicious behavior.

Conclusions

Many different technologies can be used to secure the operation of a network client. Ongoing research in sandboxing applications can be found in Prevelakis and Spinellis (2001) and Fu *et al.* (2000) while NSA (2001) investigates architectures for providing operating system security mechanisms. Firewall vendors should consider more the ease of configuration while virtual machines need to be enhanced in order to provide better performance. Code signing is an improvement in controlling software origin but the fact that it is based on human judgment poses the need to use it in combination with sandboxes. Smart cards seem to be a very promising technology for client protection. Protecting network clients becomes imperative as users rely more and more on them in order to conduct sensitive operations (e.g. e-commerce transactions). We believe that in the forthcoming years research in this area will empower their security.

References

Anderson, J. (1972), "Computer security technology planning study", *ESD-TR-73-51*, HQ Electronic Systems Division (AFSC), L.G. Hanscom Field, Bedford, MA, October, Vol. 1/2.

Anderson, R. (2002), "TCPA/Palladium frequently asked questions, Version 1.0", available at: www.cl.cam.ac.uk/~rja14/tcpa-faq.html (accessed 6 November 2002).

Arbaugh, B. (2002), "Improving the TCPA specification", *IEEE Computer*, Vol. 38 No. 5, August, pp. 77-9.

Balfanz, D. and Felten, E. (1999), "Hand-held computers can be better smart cards", *Proceedings of the 8th USENIX Security Symposium*, Washington, DC.

Chen, Z. (1998), "Understanding Java Card 2.0", *Javaworld Magazine*, March.

Cheswick, W. and Bellovin, S. (1994), *Building Internet Firewalls*, Addison-Wesley, Reading, MA.

Coleman, A. (1998), "Giving currency to the Java Card API", *Javaworld Magazine*.

Erlingsson, U. and Schneider, F. (1999), "SASI enforcement of security policies: a retrospective", *ACM New Security Paradigms Workshop*, July, New York, NY, pp. 246-55.

Erlingsson, U. and Schneider, F. (2000), "IRM enforcement of Java stack inspection", *2000 IEEE Symposium on Security and Privacy (SOSP '2000)*, Piscataway, NJ.

Freudenthal, M., Heiberg, S. and Willemson, J. (2000), "Personal security environment on palm PDA", *16th Annual Computer Security Applications Conference (ACSAC '00)*, New Orleans, LA, 11-15 December.

Fu, K., Sit, E., Smith, K. and Feamster, N. (2000), "MAPbox: using parameterized behavior classes to confine untrusted applications", *9th*

USENIX UNIX Security Symposium, Denver, CO.

Garfinkel, S. and Spafford, G. (2002), *Web Security, Privacy and Commerce*, 2nd ed., O'Reilly & Associates, Sebastopol, CA.

Ghosh, A. (1998), *E-Commerce Security: Weak Links, Best Defenses*, Wiley Computer Publishing, New York, NY.

Ghosh, A. (2001), *Security and Privacy for E-Business*, Wiley Computer Publishing.

Ghosh, A. and McGraw, G. (1998), "An approach for certifying security in software components", *21st National Information Systems Security Conference*, National Institute of Standards and Technology (NIST), Crystal City, VA, pp. 82-6.

Gong, L., Mueller, M., Prafullchandra, H. and Schemers, R. (1997), "Going beyond the sandbox: an overview of the new security achitecture in the Java development kit (JDK) 1.2", *USENIX Symposium on Internet Technologies and Systems*, Monterey, CA, December, pp. 103-12.

Gritzalis, S. and Spinellis, D. (1997) "Addressing threats and security issues in World Wide Web technology", *3rd International Conference on Communications and Multimedia Security*, Athens, Greece, pp. 33-46.

Gritzalis, S., Aggelis, G. and Spinellis, D. (1998), "Programming languages for mobile code: a problems viewpoint", *1st International Network Conference INC '98*, Plymouth, pp. 210-17.

Kozen, D. (1998), *Efficient Code Certification*, Tech. Report 98-1661, Cornell University, January.

Lampson, B. (1971), "Protection", *5th Princeton Conference on Information Science and Systems*, Princeton, NJ.

Lampson, B. (1983), "Hints for computer systems design", *9th ACM Symposium on Operating Systems Principles*, Bretton Woods, NH.

Lindhorn, T. and Yellin, F. (1997), *The Java Virtual Machine Specification*, Addison-Wesley, Reading, MA.

Lobel, J. (1986), *Computer Security and Access Control: Foiling the System Breakers*, McGraw-Hill, New York, NY.

McGraw, G. and Felten, E. (1997), "Understanding the keys to Java Security", *Javaworld Magazine.*

McGraw, G. and Felten, E. (2000), "Securing Java", Wiley Computer Publishing, New York, NY.

Martin, D. Jr, Rajagopalan, S. and Rubin, A. (1997), "Blocking Java applets at the firewall", *1997 IEEE Symposium on Network and Distributed Systems Security*, San Diego, CA, March, available at: www.cs.bu.edu/techreports/96-026-java-firewalls.ps.Z (accessed 6 November 2002).

Meyer, K., Schaeffer, S., Baker, D. and Manning, S. (1995) "Addressing threats in World Wide Web technology", *11th Annual Computer Security Applications Conference*, pp. 123-3.

Microsoft Corporation (2001) "Code signing with Microsoft authenticode", *MSDN Library Online.*

Microsoft Corporation (2002), *Microsoft "Palladium": A Business Overview*, White Paper, August, available at: www.microsoft.com/presspass/features/2002/jul02/0724palladiumwp.asp (accessed 6 November 2002).

Necula, G. and Lee, P. (1996), "Safe kernel extensions without run-time checking", *2nd USENIX Symposium on Operating Systems Design and Implementation (OSDI '96*, Seattle, Washington, DC, October.

NSA (2001), "Security enhanced Linux (SELinux)", available at: www.nsa.gov/selinux/ (accessed 20 November 2001).

Organick, E. (1972), *The MULTICS System: An Examination of Its Structure*, MIT Press, Cambridge, MA.

Prevelakis, V. and Spinellis, D. (2001), *USENIX 2001 Technical Conference*, USENIX Association.

Saltzer, J., Reed, D. and Clark, D. (1984), "End-to-end arguments in system design", *ACM Transactions on Computer Systems*, Vol. 2 No. 4, pp. 277-88.

Schneier, B. (2002), "Palladium and the TCPA", *Crypto-Gram Newsletter*, 15 August, Counterpane Internet Security, Inc., available at: www.counterpane.com/crypto-gram-0208.html (accessed 6 November 2002).

Schoen, S. (2002), "Palladium details", *ActiveWin.com*, July 8, available at: www.activewin.com/articles/2002/pd.shtml (accessed 6 November 2002).

Sommerville, I. (2001), *Software Engineering, 6th Edition*, Addison-Wesley, Reading, MA.

Stallings, W. (1995), *Network and Internetwork Security: Principles and Practice*, Prentice-Hall, Englewood Cliffs, NJ.

Trusted Computing Platform Alliance (TCPA) (2000), *Building a Foundation of Trust in the PC*, White Paper, January.

Visa International (2001), "Chip migration plan", available at: www.visa.com (accessed 10 November 2001).

VMware Inc. (2000), "VMware GSX server", available at: www.vmware.com/pdf/gsx_whitepaper.pdf (accessed 10 November 2001).

Wahbe, R., Lucco, S., Anderson, T. and Graham, S. (1993), "Efficient software-based fault isolation", *14th ACM Symposium on Operating Systems Principles (SOSP' 93*, Asheville, NC, pp. 203-16.

Wallach, D. and Felten, E. (1998), "Understanding Java stack inspection", *1998 IEEE Symposium on Security and Privacy (SOSP' 98)*, Oakland, CA, May.

Zwicky, E., Cooper, S. and Chapman, D. (2000), *Building Internet Firewalls, 2nd Edition*, O'Reilly & Associates, Sebastopol, CA.

Further reading

Gollman, D. (1999), *Computer Security*, Wiley Computer Publishing, New York, NY.