Risks, Technologies, and Strategies

Web Security & Commerce

A NUTSHELL®
HANDBOOK

Simson Garfinkel with Gene Spafford

O'Reilly & Associates, Inc.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

Web Security & Commerce

Simson Garfinkel
with Gene Spafford

O'REILLY™
Cambridge · Köln · Paris · Sebastopol · Tokyo

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

Web Security & Commerce
by Simson Garfinkel, with Gene Spafford

Copyright © 1997 O'Reilly & Associates, Inc. All rights reserved.
Printed in the United States of America.

Editor: Deborah Russell

Production Editor: Clairemarie Fisher O'Leary

Printing History:

June 1997: First Edition.

Nutshell Handbook and the Nutshell Handbook logo are registered trademarks and The Java Series is a trademark of O'Reilly & Associates, Inc. Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and O'Reilly & Associates, Inc. was aware of a trademark claim, the designations have been printed in caps or initial caps.

While every precaution has been taken in the preparation of this book, the publisher assumes no responsibility for errors or omissions, or for damages resulting from the use of the information contained herein.

This book is printed on acid-free paper with 85% recycled content, 15% post-consumer waste. O'Reilly & Associates is committed to using paper with the highest recycled content available consistent with high quality.

ISBN: 1-56592-269-7

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

- The Java language doesn't have pointers. That's good, because many C/C++ programmers don't understand the difference between a pointer to an object and the object itself.*
- Java only has single inheritance, making Java class hierarchies easier to understand. And since Java classes can implement multiple interfaces, the language supports many of the advantages of multiple-inheritance languages.
- Java is strongly typed, so you don't have problems where one part of a program thinks that an object has one type, and another part of a program thinks that an object has another type.
- Java has a sophisticated exception handling system.

All of these features combine to make Java a *safe* programming language: Java programs rarely misbehave wildly when given data that is slightly unexpected. (Instead, they simply generate an exception, which usually causes the program to terminate with a run-time error.) And because most security problems are the result of bugs and programming errors, it is thought that programs written in the Java language will be more secure than programs written in traditional languages such as C and C++.

Java Security

Java was not designed to be a secure programming language. Under Java's original vision, programs would only be downloaded by an equipment manufacturer or an approved content provider. Java was designed for a closed programmer community and for a somewhat constrained set of target environments.

When Java was repositioned for the Web, security immediately became a concern. By design, the World Wide Web allows any user to download any page from anyone on the Internet, whether it is from an approved content provider or not. If web users can download and run a program by simply clicking on a web page, then there needs to be some mechanism for protecting users from malicious and poorly constructed programs.

Safety is not security

Having a safe programming language protects users from many conventional security problems. That's because many security-related problems are actually the

* C lets you do some interesting things. For instance, if you define `char *p; int i;` in a program, you can then use the terms `p[i]` and `i[p]` almost interchangeably in your code. Few C programmers understand the language well enough to understand quirks such as this.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

result of programming faults.* Java eliminates many traditional sources of bugs, such as buffer overflows.

But a safe programming language alone cannot protect users from programs that are intentionally malicious.† To provide protection against these underlying attacks (and countless others), it's necessary to place limits on what downloaded programs can do.

Java employs a variety of techniques to limit what a downloaded program can do. The main ones are the Java sandbox, the SecurityManager class, the Bytecode Verifier, and the Java Class Loader. These processes are illustrated in Figure 3-2 and described in the following sections.

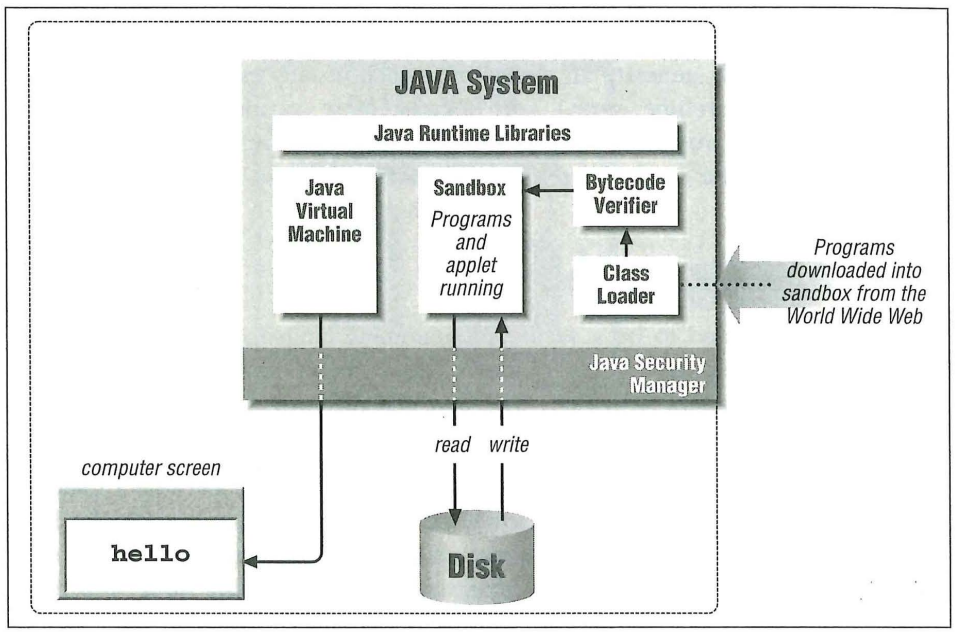

Figure 3-2. The Java sandbox, SecurityManager class, Bytecode Verifier, and Class Loader

* In technical terminology, programmers make *errors* that result in *faults* being present in the code. When the faults cause the code to produce results different from the specifications, that is a *failure*. Most casual users simply refer to all of these as "bugs," and that's why we do too.

† In fact, safety is an aid to people writing Trojan horses and hostile applications. Safety will help minimize the chances that a Trojan horse program will crash while it is reformatting your hard disk. Safety also helps ensure that the applet scanning your computer for confidential documents and surreptitiously mailing them to a remote site on the Internet won't go into an infinite loop.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

Sandbox

Java programs are prohibited from directly manipulating a computer's hardware or making direct calls to the computer's operating system. Instead, Java programs run on a virtual computer inside a restricted virtual space.

Sun termed this approach to security the Java "sandbox," likening the Java execution environment to a place where a child can build things and break things and generally not get hurt and not hurt the outside world.

SecurityManager class

If all Java programs were restricted so that they couldn't send information over the network, couldn't read or write from the user's hard disk, and couldn't manipulate the computer's input/output devices, they would probably be nearly secure: after all, there would be little damage that the programs could do.* Of course, these limitations would also make Java a much less exciting programming environment: that's because there wouldn't be much of anything interesting that Java programs could do either.

Java uses a series of special classes that allow programs running inside the sandbox to communicate with the outside world. For example, the Java class FileOutputStream allows a Java program to open a file for writing to the user's hard disk.

The creators of Java believed that programs that are downloaded from an untrusted source, such as the Internet, should run with fewer privileges than programs that are run directly from the user's hard disk. They created a special class, called SecurityManager, which is designed to be called before any "dangerous" operation is executed. The SecurityManager class determines whether the operation should be allowed or not.†

Class Loader

Because most of the security checks in the Java programming environment are written in the Java language itself, it's important to ensure that a malicious piece of program code can't disable the checks. One way to launch such an attack would be to have a malicious program disable the standard SecurityManager class or replace it with a more permissive version. Such an attack could be carried out by a downloaded piece of machine code or a Java applet that exploited a bug in the Java run-time system. To prevent this attack, the Class Loader examines classes to make sure that they do not violate the run-time system.

---

* However, the code could replicate itself and tie up processing resources, resulting in a denial of service.

† In Netscape Navigator, the java.lang.SecurityManager base class is subclassed by the netscape.lang.AppletSecurity class that implements the actual Java security policy.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

In this chapter:
- *Why Code Signing?*
- *Microsoft's Authenticode Technology*
- *Obtaining a Software Publisher's Certificate*

9

Code Signing and Microsoft's Authenticode

Code signing is a technique for signing executable programs with digital signatures. Code signing is designed to improve the reliability of software that's distributed over the Internet. It is meant to provide a system for trusting downloaded code and reducing the impact of malicious programs, including computer viruses and Trojan horses.

This chapter describes the mechanics of code signing. For a discussion of why it might not provide the safety that its backers assert, see Chapter 4, *Downloading Machine Code with ActiveX and Plug-Ins.*

Why Code Signing?

Walk into a computer store and buy a copy of Microsoft Windows 95, and you're pretty sure that you know what you are buying and who produced it. The program, after all, comes shrink-wrapped in a box, with a difficult-to-forge security hologram. Inside the box is another hologram and a CD-ROM. You know that your CD-ROM or floppy disks have the same program as every other CD-ROM or floppy disk sold in every other Windows 95 box. Presumably, the software was checked at the factory, so you have every reason to believe that you've got a legitimate and unaltered copy.

The same can't be said for most software that's downloaded over the Internet. When Microsoft released its 1,264,640-byte Service Pack 1 for Windows 95, the only way to be sure that you had a legitimate and unaltered copy was to download it directly from Microsoft's web site yourself—and then hope that the file wasn't accidentally or intentionally corrupted either on Microsoft's site or while it was being downloaded.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

What's worse, if you wanted to save yourself some time and grab Service Pack 1 from a friend, there was no way that you could inspect the file and know whether it was good or not: your friend's copy might have been corrupted on his hard disk, or it might have been infected with a virus, or it might not even be the right program. How do you know if it is a true copy, other than trusting your friend at his word?

With Microsoft's Service Pack 1, you couldn't. But starting with Internet Explorer 3.0, Internet Windows users were given a powerful system for trusting the authenticity of their software: digital signatures for executable programs.

Code Signing in Theory

Code signing is supposed to bring the assurance of shrink-wrapped software to the world of software that's distributed electronically. It does this by adding two things to an executable:

- A digital signature that signs the executable with a secret key.
- A digital certificate, which contains the corresponding public key, the name of the person or organization to whom that key belongs, and a digital signature signed by a recognized certification authority.

These are shown in Figure 9-1.

To work, code signing presupposes the existence of a working public key infrastructure. Otherwise, there is no way to tell whose signature is on a piece of signed code.

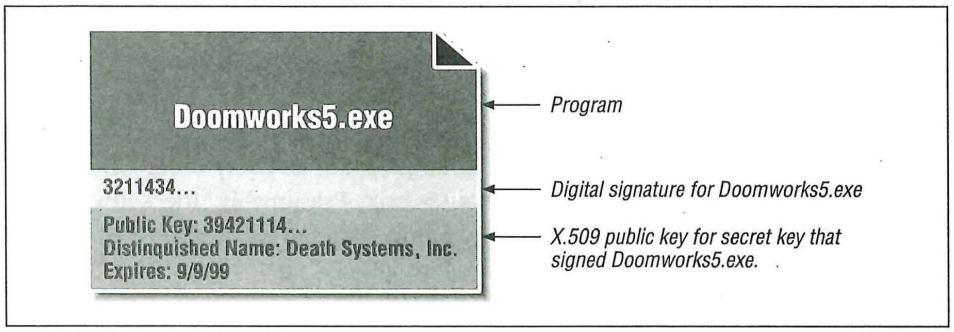

Figure 9-1. An idealized diagram of a piece of signed code, showing the code's digital signature and the corresponding digital certificate

To be useful, the signatures must be verified. Ideally, all code signatures should be verified when each piece of code is downloaded as well as before each time it is run. In this way, code signing can detect both malicious attempts to modify

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

code and accidental modifications that might result from operating system errors or hardware failure. This is because once a program is modified, the signature on that program will no longer verify. Thus, code signing can dramatically boost the reliability of today's computer systems by giving us reliable ways of detecting modifications in programs before those programs are run. Unfortunately, no operating system now offers this run-time verification of installed programs.

Code signing has also been proposed as a way of creating accountability for people who write programs and distribute them on the Internet. The idea is that Internet users should be taught not to run programs that are unsigned.* Then, if a malicious program is distributed on the Internet, it will be a simple matter to find out who distributed the program and punish them. By establishing high penalties for people who distribute malicious programs, as well as a reliable technique for tracking their authors, it is thought that the incidence of such programs will be greatly diminished.

Code Signing Today

Today there are several proposals for code signing:

- Authenticode, a system developed by Microsoft for signing all kinds of downloaded code.
- JAR, a Java Archive format developed by JavaSoft and Netscape which may be extended to support digital signatures. JAR is essentially ZIP files with digital signatures.
- Extensions to the PICS content rating system to allow organizations to explain what different kinds of digital certificates actually mean. (Chapter 17, *Blocking Software and Censorship Technology* describes PICS.)

The World Wide Web Consortium has an active code signing initiative underway. In the meantime, Microsoft seems to be going ahead with its Authenticode technology.

Code Signing and U.S. Export Controls

Unlike other technologies, there are no U.S. export controls on signed applications, on programs for verifying signed applications, or on the public keys used to validate signed applications. This is because signing does not embed secret messages in the signed documents, and it is inherently easy to examine a signature to determine if it is genuine—you simply try to verify it.

* If public education fails, system software can always be modified so that unsigned programs cannot run.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

Beware the Zipper!

The need for code signing was demonstrated in 1995 and 1996, when a program called *PKZIP30B.EXE* was reportedly uploaded to many software libraries on the Internet. The program appeared to be the 3.0 Beta release of PKZIP, a popular disk compression program. But when unsuspecting users downloaded the program and tried to run it, the rogue program actually erased the user's hard disk. That didn't exactly do wonders for the reputation of PKWare, PKZIP's creator.

Digital signatures could have prevented this mishap. If PKWare had previously adopted a policy of signing their programs with the company's digital signature, then bad hackers could never have created the *PKZIP30B.EXE* file and credibly passed it off as being the creation of PKWare. That's because the rogue hackers who distributed the *PKZIP30B.EXE* program couldn't have signed it with PKWare's digital signature.

Of course, PKWare would have to have put in place a policy of signing their programs with their secret key and making sure that the company's public key was in wide distribution. They would also have had to make sure that there was a simple-to-use verification program that was widely available.

How could PKWare have done this? It would have been tough. The company could have created a PGP key, distributed it, and used that key to sign new versions of its program. But how could anybody have trusted the key? Or the company could have built its own public key and the verification program in previous versions of PKZIP. The program could have been equipped with a "verify and upgrade" command that would inspect the digital signature of a proposed patch or upgrade, make sure that it matched, and then perform the necessary upgrade. Not only would this have thwarted the malicious hackers, it would also have also assured the company's customers that the upgrades they got from friends were true and unadulterated.

Today such bootstrapping is no longer necessary. Companies can simply obtain a software publisher's certificate from a recognized certification authority and use it to sign the programs that they distribute.

Microsoft's Authenticode Technology

Authenticode is a system developed by Microsoft for digitally signing executable code. Authenticode was publicly announced in June of 1996 as part of Microsoft's Internet Explorer 3.0 and ActiveX technologies.

ActiveX is a system for downloading programs from web pages to end user computers. There are considerable security issues associated with ActiveX. Authen-

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

- Another network scanning program is ISS (Internet Security Scanner). For some reason, the public domain version of ISS has gained a following among the computer underground, even though it is not inherently more or less powerful than SATAN: ISS simply scans for different problems. The program is made available by Internet Security Systems, Inc., which licenses a more powerful (and useful) version of the program. You can learn more about ISS at *http://iss.net/iss/*.
- SomarSoft (*http://www.somarsoft.com*) offers several tools for analyzing information culled from NT logs and databases. KSA, mentioned above, also provides analysis and integrity checking for NT environments. Likewise, some commercial virus scanning products can provide signature-based integrity checks for NT binaries and data files.

Intrusion detection programs

Intrusion detection programs are the operating system equivalent of burglar alarms. As their name implies, these tools scan a computer as it runs, watching for the tell-tale signs of a break-in.

When computers crackers break into a system, they will normally make changes to the computer to make it easier for them to break in again in the future. They will then frequently use the compromised computer as a jumping-off point for further attacks against the organization or against other computers on the Internet. The simplest intrusion detection tools look for these changes by scanning system programs to see if they have been modified.

Most working intrusion detection tools are commercial. Three systems currently on the market are:

- Stalker and WebStalker, by Haystack Labs. These systems monitor a host computer for the kinds of changes in privilege that are associated with an attack, rather than legitimate operations. For information, check *http://www.haystack.com/*.
- NetRanger, by WheelGroup, which uses network monitoring to scan for intrusions. For information, check *http://www.wheelgroup.com/*.
- Gauntlet ForceField, by Trusted Information Systems, which scans a system for the signs of attack. For information, check *http://www.tis.com/*.

Faults, Bugs, and Programming Errors

One of the biggest threats to the security of your system is the presence of software *faults* or *bugs*. These can cause your web server or client to crash, corrupt your information, or, worst of all, allow outsiders unauthorized access. However,

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

it is amazing how many organizations blindly continue the dangerous practice of continued purchase and/or use, with little or no complaint, buggy software employing methods and technology known to be at risk. Upon suffering a break-in, the purchaser* expresses shock and surprise. That same person will then later stand in line at midnight to be the first to buy the next release of the software or will rush to download the next "pre-beta" release of the software from the Net.

Initial purchase

Today there are not many choices when purchasing commercially produced and maintained software if security and responsibility are deciding factors.

Some vendors are much worse than others. However, because vendors and products are changing very quickly, we have resisted naming names. Some of the factors that you should consider include:

- Which vendors have the best reputation of producing bug-free, well documented software. Ideally, you should be able to get established metrics and test evaluations from the vendor to illustrate the quality control measures employed.
- Which vendors respond in a timely and open fashion to reports of security or performance-relevant faults in their products. Some vendors have a history of ignoring users unless there is significant "bad press" from complaints or incidents. These vendors should be avoided.
- Which vendors employ good design, with issues of security, safety, and user interface given due importance. Systems resistant to attacks and user mistakes are much better to use in situations where you need dependable operation.
- Whether you wish to use new software with more features or use "old-generation" software for which the problems are presumably well-known.

Here are some things to request or require when shopping for software and systems:

- Proof of good software engineering practices in the design, coding, and testing of the software.
- Documentation showing the results of operational and stress testing of the software and system in environments similar to your intended use.
- A written statement of the vendor's policy for accepting, documenting, and responding to reports of faults in the product.

---

* We use "purchaser" to also mean "the person who downloaded the freeware" if the software in question is noncommercial in nature.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

- A written statement of how the vendor notifies customers of newly fixed security flaws. (The most responsible vendors release notices through FIRST teams* and through customer mailing lists; the least responsible vendors never announce fixes, or bury them in lists of other bug fixes in obscure locations.)

To solve the industry's apparent inability to take security seriously is going to require some major changes. It might also require a few successful liability lawsuits. In the meantime, if you add your voice to those of others requiring better security, you not only help to raise awareness, but you should be able to protect yourself somewhat by making a better platform choice.

Note, however, that no software vendor is going to warrant its products against losses related to unsecured code—not even the vendors of security products. Typically, losses are limited to the cost of the media upon which the product is shipped.

Bugs and flaws

The Internet is a powerful tool for transmitting information. In a matter of minutes, news of the latest security flaw can be sent around the world and read by thousands or hundreds of thousands of individuals who may be eager to exploit it. Some of these individuals may attempt to break into your computer with their new knowledge. Sometimes they may be successful.

Thus, if you administer a computer that is connected to the Internet, it is important that you monitor bulletins issued by your vendor and that you install security-related patches as soon as they are made available. Most vendors have mailing lists that are specifically for security-related information.

Another source of information are FIRST teams such as the CERT (Computer Emergency Response Team) at Carnegie Mellon University. The CERT collects reports of computer crime, provides the information to vendors, and distributes information from vendors regarding vulnerabilities of their systems. Experience over time, however, has shown that CERT and many other response teams do not make information available in a timely fashion. We suggest that you monitor announcements from the response teams, but that you don't depend on them as your primary information source.

As a backup, you might also subscribe to one or two of the security-related mailing lists, such as *nt-security@iss.net* or *firewalls@greatcircle.com.*

Before you install any patch, be sure that the patch is an authentic patch provided by your vendor. You can often check a file's digital signatures and cryptographic

* Forum of Incident Response and Security Teams, the worldwide consortium of major computer incident response groups. Visit *http://www.first.org* for more information.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

checksum to determine the authenticity and integrity of a patch before you install it. Also, be very wary of applying patches found in mailing lists and on bulletin boards: at worst, they may be planted to trick people into installing a new vulnerability. At best, they are often produced by inexperienced programmers whose systems are unlike yours, so their solutions may cause more damage than they fix. Caveat emptor!

Logging

Many of the services on networked computers can be configured to keep a log of their activities. Computers that run the UNIX and NT operating systems can have their syslog system customized so that events are written into a single file, written to multiple files, sent over the network to another computer, sent to a printer, or sent to another device.

Log files are invaluable when recovering from a security-related incident. Often, they will tell you how an attacker broke in, and even give you clues to track down who the attacker is. Sometimes, log files can be submitted as evidence in a court of law. However, if people break into your computer, the first thing that they will do is to cover their tracks in your log files by either erasing or subtly modifying the files. The only way to prevent this is to set up a secured log server that will collect log entries from other computers on your network. This log server can be either inside or outside your firewall; you may need to use two of them if you cannot configure your firewall to forward the UDP packets the UNIX log system uses.

Your log server should be a computer system that offers no services to the network and does not support user accounts. The idea is to avoid having people break into your log server after they have broken into other systems on your network. Alternatively, you may have a log server that is used for other purposes, but augments its internal log file with an external write-only log device. For example, you may send log events down a serial link to another computer, such as an obsolete PC machine. This computer can simply record the log entries in a disk file and display them on a console. In the event that the log server is broken into, you will still have a record of log events on the PC machine.

You should have logging enabled for all of your servers and you should be sure that these logs are examined on a regular basis. You may wish to write a small script that scans through the log file on a daily basis and filters out well-known events that you are expecting, or use a log analyzer, as mentioned earlier. The events that are left, by definition, will be the events that you are not expecting. Once you have a list of these events, you'll either go back to modify your scripts to suppress them, or you'll make phone calls to figure out why they have occurred.

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC

Web/Security

O'REILLY™

Web Security & Commerce

Attacks on government web sites, break-ins at Internet service providers, electronic credit card fraud, invasion of personal privacy by merchants as well as hackers—is this what the World Wide Web is really all about?

Web Security & Commerce cuts through the hype and the front page stories. It tells you what the real risks are and explains how you can minimize them. Whether you're a casual (but concerned) web surfer or a system administrator responsible for the security of a critical web server, this book will tell you what you need to know. Entertaining as well as illuminating, it looks behind the headlines at the technologies, risks, and benefits of the Web. Topics include:

- User safety—browser vulnerabilities, privacy concerns, and issues with Java, JavaScript, ActiveX, and plug-ins
- Digital certificates and cryptography—how digital certificates assure identity, what code signing is about, and the basics of how encryption works on the Internet today
- Web server security—detailed technical information about SSL, TLS, host security, server access methods, and secure CGI/API programming
- Commerce and society—how digital payments work, what blocking and censorship software is about, and what civil and criminal issues you need to understand

"Garfinkel and Spafford deal head on with key elements of Internet and enterprise security. Web Security & Commerce *addresses modern security technologies and applications in a comprehensive fashion, and is an important work in the explosive, fast-moving, and highly visible security field."*

—Eric Greenberg, Group Security Product Manager, Netscape Communications Corporation

"This is a truly useful book that can help people avoid a lot of the risks in Webware. It is intelligently written, timely, informative, accurate, comprehensive, understandable, and a great pleasure to read. It is the Web-ster's definitive guide to security."

—Peter G. Neumann, moderator of ACM "RISKS" Forum and author of Computer-Related Risks

"This book is packed with useful information and solid advice for Web users, Webmasters, and developers. Garfinkel and Spafford skip the usual marketing hype and tell us how and why Web security works—or breaks down—in the real world."

—Dr. Edward Felten, head of Princeton's Secure Internet Programming and author of Java Security

ISBN 1-56592-269-7

US $32.95
CAN $46.95
90000

9 781565 922693

6 36920 92269 8

RepKover™

Printed on Recycled Paper

Exhibit 1045
Microsoft Corporation v. Edge Networking Systems, LLC