Mandatory Protection for Internet Server Software

Secure Computing Reprint Series.
Richard E. Smith

SECURE COMPUTING

Secure Computing Corporation
Corporate Headquarters
One Almaden Boulevard, Suite 400
San Jose, CA 95113-2211
tel +1.800.379.4944
tel +1.408.918.6100
fax +1.408.918.6101

European Headquarters
tel +44.1753.826000
fax +44.1753.826001

Asia/Pacific Headquarters
tel +65.535.6206
fax +65.535.8138

www.securecomputing.com

SECURE COMPUTING

Abstract

Server software on the Internet is today's high point for software at risk. Ongoing reports of security flaws suggest that conventional Internet server software packages are intrinsically vulnerable to server overrun, an attack that subverts the server's behavior and causes it to run attack code instead. The attack code then penetrates other portions of the server host or site unless there are additional defenses. Mandatory protection mechanisms, like those developed for multilevel security applications, can limit the risks of server overrun to a site. Commercial systems have been developed that use three distinct mechanisms: Unix "chroot" isolation, multilevel security (MLS), and Type Enforcement™ technology . This paper compares and contrasts these three mechanisms for server protection.

Introduction

Internet server software is both visible and vulnerable. It is visible because that is its purpose: it must have a well known address so that arbitrary clients on the Internet may reliably locate it and communicate with it. Its visibility to legitimate users also makes it accessible by potential attackers. Server software is vulnerable because it is complex software, and complex software always contains flaws. Server software in particular tends to contain security flaws because security is rarely the developers' primary concern. Developers focus primarily on refining the capabilities of the network service. In many cases security needs are rarely well understood until after a new service has been deployed for some time. Attackers know that unusual or surprising behavior can cause server software to behave unpredictably, and that the resulting behavior can manifest itself as a security flaw.

Recently, server attacks have even been in the news. World Wide Web sites belonging to the U.S. Department of Justice, the Central Intelligence Agency, and the Dole for President campaign have all been attacked. In each case, the web site was intended only as a publicity site and the web pages were dramatically changed. If the sites had been involved in Internet commerce, the penetrations might have allowed attackers to order merchandise or perhaps issue cash refunds. Such attacks are inevitable on a profit making Internet.

Sites take various steps to protect their servers from attack. A popular first step is to direct all outside traffic to a single host and to focus security measures on that host. An additional security measure is to place the outside server software on a "dual homed host," one with a network interface for Internet traffic and a separate interface for the site's inside traffic. Both Internet and inside users can reach the server host, but Internet traffic does not flow directly into the inside network. This architecture has evolved into today's "application gateway" firewalls. However, the dual homed architecture is not sufficient itself to protect the site's inside network from attack. Staged attacks can and do penetrate sites by first subverting a server and then using the server process to penetrate the inside network. The generic staged attack is illustrated in Figure 1.

Server overrun has been reported numerous times in Internet server software. The Internet Worm overran fingerd and sendmail server software to initiate staged attacks[6]. Since then, several security problem tracking organizations have emerged, including the Computer Emergency Response Team (CERT) and the Department of Energy's Computer Incident Advisory Capability (CIAC). Efforts to close detected security flaws has been documented in a variety of security advisories issued by CERT and CIAC over the past several years. Many reports have been about weaknesses in server software, including the most widely used Internet services: electronic mail and World Wide Web. The Unix sendmail program is justly famous both for its capabilities as a mail server and its tendency to contain security flaws[2]. Web server soft-

SECURE COMPUTING

ware, though comparatively new, has also demonstrated security relevantbugs as evidenced by CERT Advisory CA-95:04 and CIAC Bulletin G-20.

Internet sites use various techniques to deal with server overrun. Many simply accept the risk, hoping that attacks will strike their neighbors instead. Others try to limit their vulnerability by placing services on "sacrificial hosts" outside their security perimeter. Unfortunately, this approach does not work for electronic mail. There must always be a path for the e-mail to move between the inside and outside network. Attacks on the site's e-mail server will follow the same path, pretending to be legitimate e-mail traffic.

A more effective approach is to constrain the vulnerable server software using mandatory protection. In general, mandatory protection constrains the behavior of vulnerable software components and can not be disabled by them. The discretionary access controls provided by typical commercial operating systems are based on user identities and file permissions. Such protections can protect cooperating users from interfering from one another, but they are easily circumvented by a competent attacker.

Mandatory protection is used to unconditionally restrict access to sensitive resources in a computing system. The most common example is the user mode/kernel mode distinction widely used in processors to support multiprogramming. Another well known example is in "multilevel security" systems intended to prevent subverted software ("Trojan horses") from leaking classified information to unauthorized recipients. This is implemented in "A" and "B" level trusted computing systems evaluated by the National Computer Security Center[3].

Essentially, mandatory protection enforces a set of rules that can not be disabled or bypassed by normal operating software. Internet server software is constrained by these rules even if the attacker makes the software's process operate in arbitrary and unpredictable ways. The fundamental capability is to constrain how a given software process behaves regardless of what it attempts.

Today, three different forms of mandatory protection are applied to Internet servers: Unix "chroot()" isolation, multilevel security (MLS), and type enforcement. The following sections examine how each of them addresses the following eight security criteria. The criteria are chosen to illustrate differences between these approaches when protecting a site's inside network after an outside server process suffers an overrun.

• Keeps servers separate

An attack that overruns one server may be able to overrun other servers from "inside" the system unless there are explicit mandatory protections between server processes.

• Provide read only protection to selected data

This blocks any attempts to modify data files or even executable program files on the system, even if the server process is overrun.

• Lacks known vulnerabilities

Security mechanisms should not contain known, built-in vulnerabilities. While it may be possible to patch such vulnerabilities, there is less assurance that the system provides the needed level of security.

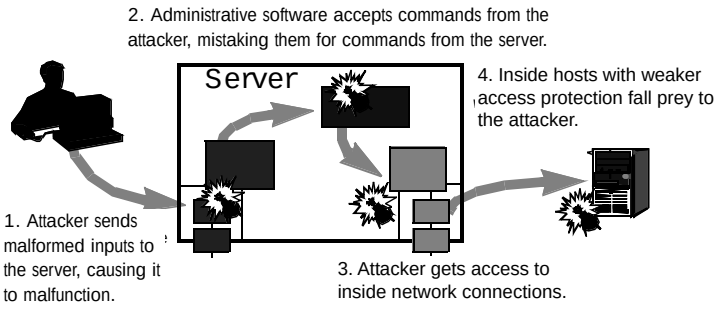

Figure 1: A staged attack penetrates a dual homed gateway in four steps.

SECURE COMPUTING

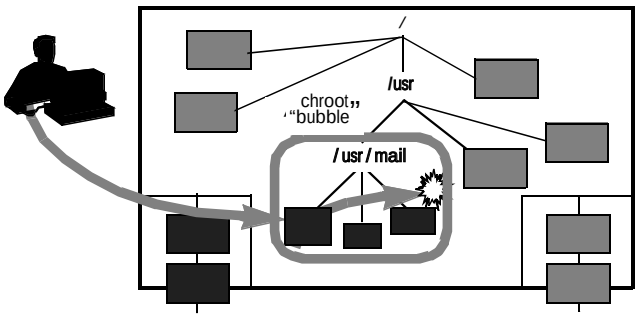

Figure 2: chroot() constrains the server to run within a "bubble" in the file system.

• Isolates outside and inside networks

A dual homed host should prevent traffic arriving at the outside network interface from leaking to the inside network interface. Traffic should move between inside and outside under the control of security software that is independent of the server software. Mandatory protection should enforce network separation and not allow direct access to the inside interface from the outside server's process.

• Security software independence

Mandatory protection should be enforced even if errors are present in the potentially fragile server software. The system's security should never depend on the correct behavior of server software.

• Blocks code insertion

Staged attacks often attempt to install custom software that serves as a platform to extend the attack to other hosts. A mechanism that blocks code insertion increases the work factor for an attacker, and reduces the likelihood of success.

• Blocks privileged code insertion

Some systems have a separate mechanism to control the insertion of privileged code. It increases the attacker's work factor if the attacker can not install privileged code.

SECURE COMPUTING

• Controls all security critical software

Mandatory protection should not be disabled when running security filtering or administrative software. Special privileges open the risk of giving an individual function more capabilities that it really needs, increasing the risk that errors or attacks can apply the software incorrectly. A given administrative function might work safely when applied to e-mail and be a cracker's tool when applied to CGI files.

Protection with chroot()

chroot() is arguably the most widely available mandatory protection mechanism because it is present in modern versions of the Unix operating system[1]. The mechanism literally changes the "root" of the hierarchical file system as seen by a process. When invoked, chroot() accepts a directory name as an argument and establishes the given directory as the new "root." Files on the system can be accessed by name only if their directory entries are within this "bubble" of files below the newly established "root." This is sketched in Figure 2.

Subsequent calls to chroot() can not reverse the original call, since there is no way to reference the original "root" directory as an argument.

The following criteria note some vulnerabilities in chroot(). These vulnerabilities are part of standard Unix behavior and should be expected to exist in standard commercial versions of Unix. It should be possible to individually patch these vulnerabilities in customized versions of Unix tailored for high security applications. However, there is no guarantee that all potential vulnerabilities have been identified.

• Keeps servers separate? Yes.

Separate "bubbles" may be established for separate services. Thus, a successful overrun of one service does not provide instant access to the others, unless the attacker exploits a weakness in chroot() itself.

• Provide read only protection to selected data? No.

chroot() only controls whether or not a file is visible to the process. If the file is visible, then it provides no additional control over how that file is used.

• Lacks known vulnerabilities? No.

The server may access files outside the "bubble" if descriptors for the files are accessible from the server's address space. The server may access other system resources, like network sockets, that are not accessed through the directory system. If the attacker's process is operating as the system "root" user, the attacker can create access paths to device drivers to get "back door" access to the file system. Some vulnerabilities could be closed in a customized Unix kernel.

• Isolates outside and inside networks? No.

chroot() does not prevent access to network resources, including the inside network interface. This vulnerability could be closed in a customized Unix kernel.

• Security software independence? No.

The server software is directly responsible for setting up its security protection. Thus, if there is an error in setting up the "bubble" there is not external agent to detect this fact and limit the damage of a failure.

• Blocks code insertion? No.

The only protection is to hide vulnerable portions of the file system from the potentially subverted process. There are no additional restrictions on accesses to files visible inside the "bubble."

• Blocks privileged code insertion? No.

Privileged code is no different from unprivileged code in this context.

• Controls all security critical software? No.

Administrative software operates outside the chroot() bubble, so it can not apply restraints to it. chroot() does not appear to be flexible enough to adequately control all software on a system.

Protection with MLS

Multilevel security (MLS) systems were developed to prevent sensitive data from flowing into regions of a computing system where they could be read by individuals not authorized to see them. Access is enforced according to the classic Bell-LaPadula model. Some MLS vendors, including AT&T and SecureWare, are now applying MLS to the protection of commercial server software. At least one Internet firewall product, the Harris Cyberguard, also uses MLS as a mandatory protection mechanism.

Figure 3 illustrates the application of MLS to an Internet server. The example shows three separate domains operating three sets of separately labeled processes. The "inside" and "outside" domains carry labels that are mutually incomparable. This prevents processes in either domain from reading or writing to files in the other domain. The label for the third "trusted" domain dominates the other two. Internet server software operates in the "inside" and "outside" domains. Trusted software in the "trusted" domain moves traffic between the servers in the outside and inside domains.

• Keeps servers separate? Yes.

The MLS mechanism can enforce mandatory separation between servers if additional incomparable labels are defined for the individual servers. In this context, the server specific labels are not being used as confidentiality labels, and they typically map onto TCP/IP port numbers associated with specific services. If an attacker penetrates a particular server, the MLS protection prevents access to other server processes or data. The only access to other servers remains the "front door" through each servers' protocol mechanisms.

• Provide read only protection to selected data? Yes.

Separate security levels can be used to make some data items read only. The Bell-LaPadula access rules state that data at a lower level can not be written by a process at a higher level under normal circumstances. MLS based systems can place executables and other data files that need write protection at "low" security levels and run servers at higher levels.

• Lacks known vulnerabilities? Yes.

Unlike chroot(), MLS has no defined-in vulnerabilities. The basic concepts have been rigorously analyzed as part of several high assurance system development programs according to the published standards [3].

• Isolates outside and inside networks? Yes.

An MLS system can be configured so that the inside and outside networks are separated by a mandatory protection boundary.

• Security software independence? Yes.

The mandatory protection boundaries are defined by the lattice or graph, and not by the server software. The protections will be enforced regardless of any flaws in the server software.

• Blocks code insertion? No.

The MLS mechanism does not generally distinguish between executable access to files and other forms of access. Thus, if there are writable file components available to a server process (e.g. for queues) then a subverted server process could install subverted software in the file system.

• Blocks privileged code insertion? Yes.

While generic MLS mechanisms do not control insertion of executable files, many MLS systems block "untrusted" software (e.g. Internet servers) from installing software that may then be executed in a more privileged mode.

• Controls all security critical software? No.

MLS systems always include special privileges that bypass the mandatory protections. A dual homed server needs to use a privileged process to move data from the security context of one network to

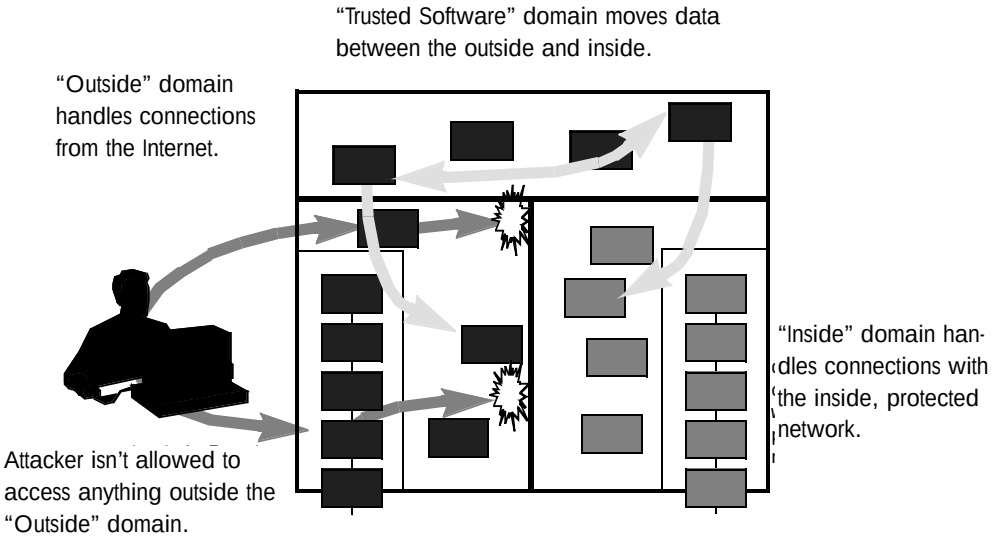

Figure 3: Multilevel security (MLS) mechanisms can be used to shield inside networks and separate network services from outside attack.

SECURE COMPUTING

the other's. Administrative software also needs privileges in order to modify configuration and permission files that the servers access in a "read only" mode. The permissions are generally very coarse: if a process has write access to the next level down, it has write access to all levels below it and can not be further constrained.

Protection with Type Enforcement™ technology

Type Enforcement technology was originally developed for the Logical Coprocessing Kernel (LOCK ®) [4], a research project that evolved into the Secure Network Server (SNS) [5], and is now commercially available in the Sidewinder ™ firewall server, which uses it to protect a variety of services including e-mail, FTP, and World Wide Web [7].

Type Enforcement technology is similar to MLS except that access permissions have finer grained control. All processes in the system are assigned to domains and all data objects are assigned types. All accesses allowed to a process in a particular domain are specified in a set of type enforcement databases. If processes in a given domain are not explicitly granted permission to create, read, write, execute, or destroy data objects of a given type, then the system blocks them from doing so. It is possible to construct an MLS lattice using a type enforcement database.

Server protection is similar to the MLS case. Processes in the "outside e-mail server" domain are allowed to listen() to the e-mail socket of the outside network stack, granted read/write access to the "outside mail queue" data type, and granted execute access to the "e-mail server program" data type. Processes in the "inside e-mail server" domain have access to corresponding inside data and network types. E-mail flows between the two domains using a separate process running in the "e-mail filter" domain. Neither e-mail server has direct access to the opposite network, nor to other servers. Attacks on a server on one domain yield little useful access to other servers or other network resources, as shown in Figure 4.

• Keeps servers separate? Yes.

Servers are assigned separate domains. Access

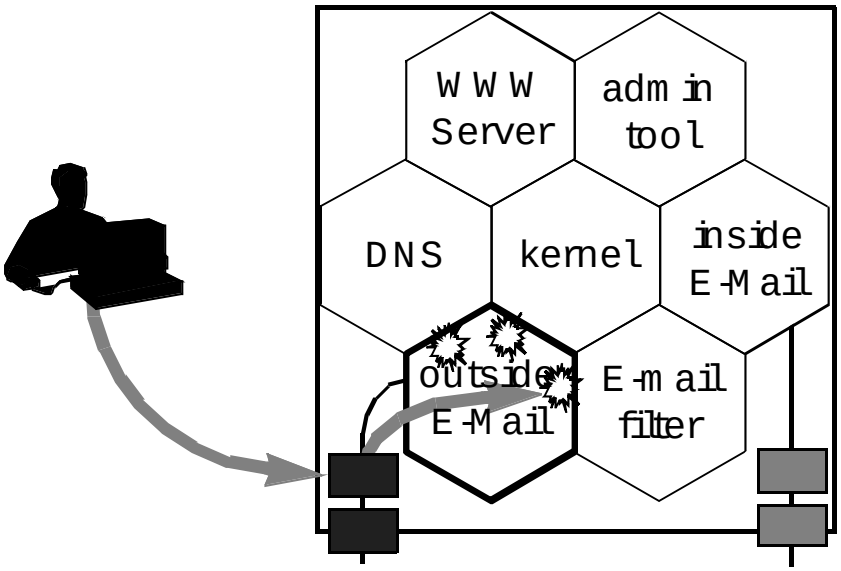

Figure 4: Type enforcement encapsulates sets of processes according to their access requirements, limiting how far an attacker can go.

SECURE COMPUTING

between domains is restricted so that an overrun in one domain does not compromise another. On Sidewinder, ports on each network are assigned types, just like other data objects.

• Provide read only protection to selected data? Yes.

A given domain can be granted specific modes of access to data items of different types. Read only data can be assigned a type that is only readable from server domains.

• Lacks known vulnerabilities? Yes.

Type Enforcement technology contains no known, built-in vulnerabilities. The concept has been rigorously analyzed as part of both the LOCK [4] and SNS [5] programs.

• Isolates outside and inside networks? Yes.

Inside and outside network interfaces are assigned different data types. Inside and outside servers are assigned different domains. Isolation is provided by denying outside domains access to the inside network, and vice versa.

• Security software independence? Yes.

The mandatory protection boundaries are defined by the type enforcement database, and not by the server software. The protections will be enforced regardless of any flaws in the server software. The Type Enforcement database can not be changed while the system is in normal operation.

• Blocks code insertion? Yes.

This is a function of the Type Enforcement database and is tailored to specific applications. In a high integrity application like Sidewinder, network server domains are never granted the ability to create executable files.

• Blocks privileged code insertion? Yes.

Privileged and unprivileged code insertion is controlled in the same manner.

• Controls all security critical software? Yes.

SECURE COMPUTING

All accesses are subjected by type enforcement checks, with no exceptions. Different software components are run in different domains, each with its distinctive access permissions to critical system data. Access requirements are specified to a fine enough degree that special "bypass" privileges are unnecessary.

Conclusions

Server software that faces an actively hostile environment needs strong protection. Typical security measures on commercial platforms were never intended to stand up against sophisticated attacks, and these measures have been shown to fail in practice. Any of these mandatory protection mechanisms will provide significantly better protection from attack. Sites that are clearly at risk of attack should take advantage of these measures. Vendors seriously committed to the development of secure, robust Internet commerce systems need to incorporate mandatory protection in their products.

chroot() provides useful, if limited, protection to server software. It may be the only mandatory protection available when services must be hosted on a generic commercial platform. Its protection can be improved if the developer is able to modify the Unix kernel source code to block its known vulnerabilities. Such changes could close known vulnerabilities and provide more robust separation between networks.

MLS protection was not explicitly designed to constrain attacks on network server software. However, its robustness in this application indicates the fundamental value of mandatory protection mechanisms. The principal shortcoming is that the access control is relatively coarse, and processes require excessive privileges if they must traverse protection boundaries. So far, there have been no public reports of MLS based firewalls or servers being compromised.

MLS and type enforcement are very similar, since type enforcement was developed as part of a high assurance MLS system. However, type enforcement had a different objective in mind and provides a finer degree of access control. Secure Computing's experience with Type Enforcement technology has shown it to be effective in the face of ongoing server attacks [7].

Originally MLS systems were intended to provide "foolproof" security protection: any program a user could install would be constrained by the security policy. Unfortunately you can not construct a "foolproof" server system with any of these mechanisms. Careful, security conscious design is vital when any of these mandatory mechanisms are applied.

Acknowledgments

This paper was inspired by a lively discussion on "Firewalls," an Internet mailing list hosted by Brent Chapman of Great Circle Associates (the mailing list is accessible through "majordomo@greatcircle.com"). I particularly appreciated the commentaries contributed the following "Firewalls" participants (in no particular order): Anton J. Aylward, Peter da Silva, Jeromie Jackson, Marcus Ranum, Darren Reed, Mark Riggins, and John G. Thompson.

Jim Riordan of Secure Computing Corporation performed penetration studies on chroot() and demonstrated several ways to prick its bubble.

References

[1] AT&T, System V Interface Definition, AT&T, Indianapolis, IN, 1986.

[2] William R. Cheswick and Steven M. Bellovin, Firewalls and Internet Security, Addison-Wesley, Reading, MA, 1994.

[3] NCSC, "Trusted Computer System Evaluation Criteria," DOD 5200.28-STD, National Computer Security Center, Ft. Meade MD, 1985.

[4] O. Sami Saydjari, Joseph M. Beckman, and Jeffrey R. Leaman, "LOCK Trek: Navigating Uncharted Space," Proceedings of the IEEE Symposium on Security and Privacy, Oakland CA, May 1989, page 167.

[5] Richard E. Smith, "Constructing a High Assurance Mail Guard," Proceedings of the 17th National Computer Security Conference, Baltimore, MD, October 1994.

[6] Eugene H. Spafford, "The Internet worm program: an analysis," Computer Communication Review, 19(1): 17-57, January 1989.

[7] Dan Thomsen, "Sidewinder: Combining Type Enforcement and Unix," Proceedings of the Annual Computer Security Applications Conference, New Orleans, LA, December 1995.

SECURE COMPUTING