

Intrusion Detection Preliminaries: Sanitizing Your E-Commerce Web Servers

Marc Myers 2000-03-31

Introduction

Intrusion Detection involves detecting unauthorized access and destructive activity on your computer system. Intrusion Detection is a clear requirement for all e-commerce merchants. According to the annual [study](#) released March 22, 2000 by the Computer Security Institute and the FBI, 90% of the survey respondents detected a computer security breach within the last twelve months. The study showed that the most serious financial losses were caused by activities that concern e-commerce merchants directly: theft of proprietary information (e.g., stealing customer credit card numbers), and financial fraud (e.g., setting up a bogus storefront).

For e-commerce merchants, the focus of Intrusion Detection is on the web servers, and their associated database management systems. E-commerce requires that the web servers communicate quickly and accurately with large databases of product and customer information. To optimize performance, these critical databases are, in most cases, placed on the same network segment as the web server, or even on the web server machine itself. For malicious hackers, this is a tempting prize. For hard-core cyber criminals, these databases are paydirt. They will break in to the web server, gain administrator-level access, locate the database, and then go to work on breaking into the database and downloading customer information.

This *does* happen. As a matter of fact, it happens more often than most of us will ever know, because the merchants who suffer break-ins often do not report them, or they report them to law enforcement agencies who do not publicize information while cases are under investigation. According to an Associated Press [report](#) released March 24, 2000, "Two 18-year-old boys were arrested in Wales, United Kingdom, on charges of breaking into electronic commerce Internet sites in five countries and stealing information on 26,000 credit card accounts, the FBI said today." Such reports cause me to wonder how many such exploits are *not* being caught. And one can only marvel at the use of the term "boys". Why is an 18 year-old who commits armed robbery a "man", and one who violates the financial integrity of 26,000 innocents a "boy". The young men who probably spent many months planning and executing this crime are not seen as *real* criminals, just misguided youth. This seems to be a naive assumption.

Setting up the most secure website possible is the social, and potentially legal responsibility of

every e-commerce merchant who either solicits, processes, or stores confidential customer information. Further, and perhaps more convincing, a secure website is also a business imperative. There is no quicker way to lose customer confidence than to lose their credit card information.

But I'm in the business of selling products, not configuring complex web technologies...

This is true. For this very reason, many e-commerce vendors have outsourced the majority of their web technology headache to third party providers. Service providers, such as [Exodus Communications](#) and [Pilot Network Services](#) are doing a brisk business by taking on the significant responsibility of providing secure, fast and highly configurable web server centers. You can turn it over to them, and concentrate on your core competencies. Indeed this may be the future direction of most e-merchants.

But...there are certainly some advantages to doing it on your own. One of the biggest is the ability to set up your system according to your business and technical requirements, without any externally imposed limitations. For example, you may want to provide a state-of-the-art database search system that uses Java Servlet technology to optimize server-side processing. There is a good chance that a service provider will not be able to offer this kind of bleeding-edge technology.

Another advantage to going it on your own is that it *forces* you to build in-house web technology expertise. Having people in the shop who understand web technology is, in my opinion, invaluable, especially if you see e-commerce as a major source of your ongoing income. Nowadays, it's the *technologists* who often come up with the ideas and innovations that make your offering stand out.

The rest of this article assumes that you've decided to tackle at least some percentage of your web server security configuration. Certainly, if you are a large operation, it is an option to outsource some of your satellite or auxiliary sites that don't require frequent updates or which have standard technology deployments, giving in-house staff more time to concentrate on your primary "mover and shaker" sites.

Ground floor security: file integrity verification

The first and most basic step is to establish a snapshot database of your file system, so that

you can, on a regular basis, compare this snapshot with the current state of the file system. If any unexpected files or directories show up, there is a possibility that the system has been breached. By performing this comparison on, say, an hourly basis for your website files and directories, and on a daily or twice-daily basis for your system files and directories, you will be able to detect and correct abnormal conditions on your servers. The best known product for automating this process is [Tripwire](#).

Certainly there are more "dynamic" products which will catch some intruders *before* they begin to alter your file systems. But they are not as foolproof as file integrity verification. To my knowledge, even the best dynamic intrusion detection software can only detect attack signatures that it has been pre-programmed to recognize.

The first step is to make sure that you know exactly what your file system is *supposed* to look like. Tripwire compares 14 distinct file attributes for UNIX systems, and 24 distinct file attributes for NT systems. If new or altered files start showing up in your file system, a "root kit" may have been installed by a malicious hacker.

A root kit is a set of tools that an intruder will use to perpetuate their stay on your system, by hiding their activities and the changes they are making to your system. Specifically, they will replace key tools that you might use to detect them, with hacked copies which will leave them unnoticed. For example, on UNIX systems, the 'ls' program is almost always replaced by current root kits, so that the hacked 'ls' will not report the files and directories created by the intruder. Another common component of a root kit is a program or set of programs to regularly erase from the log files all evidence of the intruder's activities. And another clever feature is flipping (on UNIX systems) the SETUID and SETGID permission bits on one of the hacked system programs (mode 6555), such as 'ps' or 'ls', so that when the intruder runs one of these programs, root access is established.

The good news is that any good file integrity verification program will discover a root kit immediately. You can get rid of it by going back to your latest good backup, or by reinstalling the compromised software. You should make your first snapshot of the file systems when your server is newly configured and prior to its ever being connected to the internet.

The bad news is that no amount of verification will help if your system was already compromised before you made your first snapshot of the file systems. In other words, for file integrity verification to work, you have to *start* with a clean system that you are *sure* has not

been compromised. Just because nothing out of the ordinary has happened yet, that doesn't mean you have a clean machine. An intruder might be biding their time before using the access that they have developed to your system, or they may be patiently extending their reach to other systems on your network before revealing the goal of their attack.

Most e-commerce vendors are under tremendous pressure to deploy their sites with maximum availability and minimal development time. My guess is that most e-commerce vendors did not install a file integrity verification system prior to putting their servers on the internet. This makes your task more difficult.

Yes, I DID install a file integrity verification tool before going live...

Bravo! Now you just have to make sure you use it. I recently read about a company that suspected they were under attack. When they went to run a comparison against their Tripwire baseline database, they realized it had been six months since they had updated the Tripwire database. It still helped them verify the intrusion, but they would have gotten much cleaner output from a recently updated database.

As I mentioned above, run your verifier at least once an hour, and keep track of *permitted* file system changes, so that you can update the database on a daily basis. This way it is easy to keep the file system and the database in synch.

No, I did NOT install a file integrity verification tool before going live, but now I want to...

This is where you either have to do some major reconfiguration, or ask for a minor miracle from your system administrator. Let's start with the first option, reconfiguration:

I like this option best, because there is little room for error. The basic concept is that we are going to rebuild the machine from the ground up, get the integrity verification system in there this time, and then get back on the internet. There are some ways to do this that will minimize or eliminate downtime.

One way that was suggested to me by Brian Robison of Tripwire which I think is very practical, is to use this as an opportunity to increase your capacity *and* install file integrity verification. I expanded on Brian's suggestion to develop the following example: Let's say that you have four physical servers connected to the internet - an http server, an ftp server, a mail server, and a

DNS server. Use the following round-robin approach to rebuild your system with no downtime:

1. get a new server machine, brand new
2. back up all data files on all your web servers
3. with the new server build a fifth, clean offline server to take the place of your online ftp server
4. create a file integrity database snapshot of the clean offline ftp server
5. replace the old ftp server with the clean ftp server
6. upgrade the old ftp server (memory, disk space, etc.), format the hard disk and reinstall the operating system
7. build the old ftp server into the the new mail server
8. create a file integrity database snapshot of the clean offline mail server
9. replace the old mail server with the clean mail server
10. and so on, until all servers have been reconfigured

At the end, you will have an extra server, which you can clean, upgrade, and then use to increase the capacity of your http service, or any other use that you deem appropriate. You can now begin monitoring the output of your file integrity verifications with confidence that your baseline database snapshot is 100% clean.

Now I'm worried about the baseline snapshot database - what if hackers alter, remove or replace it?

That's a valid concern. Losing that snapshot is like losing the key to your safe deposit box: not good. Needless to say, make backups on multiple media types and keep them in different physical buildings, preferably with one copy in secure storage. Tripwire encrypts their data files with 1024-bit keys, so they can not be successfully altered or replaced, though they can be removed.

Beyond this, make sure that your file integrity verification software itself has not been replaced with a Trojan Horse program. One way to do this is to install one copy of the file integrity verification software on each of your web servers and have them check *each other*. Also, you can always check the software against a clean copy from the original read-only CD, but this is a slow, manual process.

I don't have the time, money, or inclination to reconfigure my servers and install integrity verification software. My sysadmin will have to handle this...

OK, time to ask for that minor miracle. Basically, this involves a painstaking analysis of all your web servers and all other machines on the web server network segment. You need to identify files and directories that have been removed, replaced, or altered; to discover new, suspect files and directories; to monitor system logs for suspicious activity; and to vigilantly inspect process status information (and to boldly go where no man has gone before). It requires a *highly* skilled system administrator who is also very well acquainted with the day-to-day state transitions of the system.

This said, there are some good resources available to assist with this process, and, over time many of these activities become "second nature" to skilled administrators.

We can divide the task into three primary areas: the file system, the system logs, and the process status information. In all cases we are hunting for anomalies - for files, directories, log entries, and processes that *look* wrong. Then, if we are suspicious about an object, we need to identify the tools that will help us determine if there is really a problem.

File system

Malicious hackers will often install Trojan Horse files in place of standard utility programs, usually to help hide the evidence of their intrusion. For example, when looking at the file system on a UNIX box, you may notice that the 'ls' program seems to have an unusually large file size or an odd timestamp. You should then compare it with a clean copy of 'ls' from another machine or from installation media. A good tool for such quick comparisons is [MD5](#), which makes cryptographic checksums of the target files. If there appears to be a problem, then you can run the suspect 'ls' program through a debugger such as gdb, and often find out what the Trojan Horse 'ls' was designed for. Typically, a Trojan Horse 'ls' will not display any of the files or directories installed by the intruder, and may also have the SETUID and SETGID bits set to mode 6555.

Other programs commonly replaced with Trojan Horses on UNIX systems according to the [CERT Intruder Detection Checklist](#) include login, su, telnet, netstat, ifconfig, ps, find, du, df, libc, sync, and any binaries referenced in /etc/inetd.conf. Another important file to check is the etc/passwd file for new, unauthorized entries.

Using the 'find' command on UNIX, search for hidden files and directories. Note, that the 'find' command could itself be part of a root kit, rendering its output useless. After verifying that you

have a clean copy of 'find', first look for suspicious hidden files and directories:

```
find / -name ".*" -ls
```

Another useful approach is to check for files that have changed recently. The following command checks for files that have changed in the last three days. Note that this command is ineffective if the cracker has also changed timestamps on Trojan Horse files.

```
find / -mtime -3 -ls
```

You also should check, on UNIX systems, for files that do a SETUID or SETGID to give hackers unlimited access to systems. Here is how to find files that SETUID or SETGID.

```
find / -user root -perm -4000 -ls
```

```
find / -group kmem -perm -2000 -ls
```

System and Web Server logs

A proficient hacker will not want to leave any evidence behind of their activities in system logs, and will try to delete or alter revealing log entries. To get good information from log files requires a carefully designed strategy, because not only are they routinely "cleansed" by intruders, log files tend to be difficult to work with because of the voluminous amount of information that is collected.

For this reason, check logs on a daily basis, so that the amount of data does not become overwhelming. While the task may be initially daunting, soon you will start to recognize all of the normal entries and become faster at identifying anomalies.

[CERT's recommendations](#) for inspecting system and network logs specify six types of operating system-independent logs: user activity, process activity, system activity, network connections, network traffic monitoring, and program-specific logs. A review of their article will give you a general idea of what to look for.

Web server logs are also important to inspect. Typically the web server will generate an *access log*, which records all web server accesses of any type, and an *error log*, which records all failed access attempts and other errors.

For e-commerce vendors, the intruder is probably going to be gunning for your database. So become well acquainted with your database logs and review them frequently for multiple failed login attempts, for large, wildcard-style queries, and for any activity initiated by a user with direct access to the database.

Lincoln D. Stein's book 'Web Security' presents an excellent approach to monitoring UNIX log files and web server log files which I have drawn from for the following recommendations:

For UNIX logs:

1. Redirect your syslog entries to a secure remote machine or to a printer port, so that intruders will not be able to delete or alter log files.

This is done by adding entries to the `/etc/syslog.conf` file, such as:

```
*.notice @host1.yourorg.com
```

```
*.notice @host2.yourorg.com
```

These two lines cause all messages of priority "notice" or higher to be forwarded to machines host1 and host2.

2. Customize `/etc/syslog.conf` to control what gets logged. For example, log all messages from the login server, but quench all messages, except for errors, from the printer daemon.
3. Write scripts or use a log file analyzer such as [Swatch](#) to help organize and highlight suspicious log entries.

For Web Server logs:

1. Examine all access log entries with a status code of either 401 (unauthorized) or 403 (forbidden).
2. Examine all access log entries with extremely long URL's or with URL's containing non-standard characters.
3. Examine error log entries for the following errors and error types:

File permissions deny server access

This can indicate that someone is trying to get to documents outside of the web server's tree.

Password mismatch

This is caused by a user trying to access a protected document and typing the wrong password. Look for multiple consecutive errors of this type.

Client denied by server configuration

Someone has been denied access to a directory because their IP address did not match a list of allowed addresses.

Malformed header from script

A CGI script is producing bad output. The script could have been hacked, or it may just be buggy.

4. Redirect web server logs to a printer or terminal emulator.

This is done by modifying the httpd.conf file to replace the name of the log with the a reference to the device, such as:

```
ErrorLog /dev/ttyS0
```

Process status information

A common hacking technique is to break into a system, install a packet sniffer on the network, and then search all packets for login and password information. This way, the reach of the intrusion can be quickly extended to other systems, using stolen logins and passwords. The challenge is to find such sniffer programs, especially when the programs you use to query for active processes may have been replaced with Trojan Horses that will hide evidence of the sniffer program. So the first step is to verify, by comparing against installation media or another trusted source, that your process status tools are valid - in the case of UNIX I am referring to the 'ps' command.

The installation of the sniffer or any other intruder program might be caught using one of the logfile search techniques described in the previous section. But if programs installed by the intruder slip past your logfile inspections, then you can still find evidence of them using process

status information.

CERT provides a detailed [paper](#) on the use of the Solaris version of the 'ps' program to detect intruder processes.

Another useful clue, in the case of sniffer programs, is to check whether the network interface card has been placed into *promiscuous* mode. This mode causes it to report all packets that traverse the network segment, rather than just the packets explicitly addressed to its host, giving the hacker much better odds of discovering user id's and passwords.

Summing it all up

No pun intended, this is the best approach for file integrity verification: let an automated cryptographic checksum program "sum it all up" for you, many times a day, to make sure that all files match the originally installed sources. Before this is possible, though, you have to establish a squeaky clean baseline database by systematically sanitizing and baselining all the servers connected to the internet. Otherwise your checksums will only be confirming the altered state of your system, not its good health.

For e-commerce merchants, this requirement is particularly critical. A compromised system is not just a nuisance, it is also a threat to your customers' private financial information, to your business reputation, and a possible cause for legal action against you. Take the time to sanitize and reconfigure your web servers so that you can quickly and accurately verify the integrity of your system. Then you will be ready to install more sophisticated intrusion detection mechanisms that will often stop malicious hackers *before* they get to your file system.

Marc Myers is an internet security process engineer and senior consultant for [Client/Server Connection](#), a software and consulting company located in New York.

Relevant Links

[Tripwire](#)

Tripwire, Inc.

[MD5](#)

COAST - Security Archive

[Inspect your system and network logs](#)

CERT Coordination Center

[Privacy Statement](#)

Copyright 2006, SecurityFocus