

;login:

THE MAGAZINE OF USENIX & SAGE

April 2003 • volume 28 • number 2

inside:

SECURITY

Chuvakin: Ups and Downs of UNIX/Linux Host-Based Security Solutions

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

ups and downs of UNIX/Linux host-based security solutions

An exciting area of UNIX, security presents many challenges not resolved by a single technology. Host-based security solutions occupy a crucial place between network-based defenses (e.g., firewalls and network intrusion detection) and good administrative practices (e.g., securely configured and hardened hosts). Many of the SANS Top 20 Vulnerabilities can be successfully mitigated using host-based security. For example, buffer overflows in Sendmail, BIND, and RPC services can be secured using kernel-call tracing, and the consequences of most of the other attacks from the list can be discovered using file-system integrity checking.

Host-Based Security Technologies

This paper deals with UNIX host-based intrusion detection and prevention. We will take a look at UNIX host security solutions, with the focus on their inadequacies and ways to overcome them (whenever possible). Currently, many different technologies – integrity checking, kernel- and system-call tracing, log analysis, local-host NIDS – are lumped together into host-based security.

Integrity-checking software examples include Tripwire, AIDE, and a large number of lesser-known scripts and applications. Linux RPM also provides some integrity-checking functionality (“rpm -V” mode). The common feature of integrity checkers is that they keep a record of file properties such as modification or change times, location on disk, permissions, owner, and other attributes. They also compute and keep cryptographic checksums of the file contents. The simplest application of this kind would be a shell script similar to the following:

```
#!/bin/sh
#primitive integrity checker

ls -laRi / >/home/files
cd /usr/bin
ls | xargs -i md5sum {} > /home/sums

diff /home/files /home/files.old
diff /home/sums /home/sums.old
```

HIDS such as Entercept operate on a kernel level. While high-security kernel patches (such as Linux LIDS and Solaris Pitbull) also work in kernel space, they are more accurately described as “prevention” technology as opposed to “detection,” since they restrict the actions of users and applications based on certain pre-defined access control lists. Some Linux kernel modules (such as StMichael and StJude – <http://sourceforge.net/projects/stjude>) occupy an intermediate space: They can detect malicious kernel modules and can also prevent some of the damage caused by them.

The main distinction to be found in UNIX system log analysis tools is between real-time tools (that read log records as soon as they are produced, i.e., written to disk) and

by Anton Chuvakin

Anton Chuvakin is a senior security analyst with a major information security company. His areas of infosec expertise include intrusion detection, UNIX security, forensics, and honeypots. In his spare time he maintains his security portal at <http://www.info-secure.org>.



anton@netForensics.com

Host IDS usually presents a bigger administrative and management challenge than network intrusion detection.

cron-based tools (that run periodically and process accumulated log records). Some commercial HIDS (such as Dragon Squire) also have capabilities to analyze system logs. In addition, there are many freeware tools that can be used to detect intrusions using log files. Swatch is the best-known real-time tool, while logcheck, logwatch, and many others provide periodic log assessment.

Network IDSes that are used to monitor traffic only destined for a specific host (also sometimes called hybrid IDS) will not be considered here since they are much closer to network intrusion detection systems than to host security. However, if such a “hybrid” IDS analyzes network traffic after processing by the host protocol stack (e.g., at the application level), it can do many things that normal network IDS cannot do, such as analyze encrypted traffic (SSH, SSL, IPSec VPN, etc.). In addition, many of the insertion and evasion attacks described in the Ptacek and Newsham paper (“Insertion, Evasion, and Denial of Service: Eluding Network Intrusion Detection”) will not work against such IDS, since raw traffic processing is done by the target system stack and not by some other host (with its own network stack peculiarities) sniffing the traffic.

HIDS Challenges

What are some of the advantages of host-based intrusion detection products? The key difference is that while network IDS detects potential attacks (which are being sent to the target), host IDS detects attacks that succeeded, thus having a lower false positive rate. Although some might say that network IDS is thus more “proactive,” host IDS is effective in the switched, encrypted, and high-traffic environments that present certain difficulties to NIDS.

Now it’s time to turn to HIDS challenges.

First, at least some of the host IDS components are deployed on an attacked host. If the attack succeeds, the intruder will usually have “root” access to the box and might be able to disable or deceive the IDS. The usual countermeasures are “hide” (prevent an intruder from seeing the IDS and continue operation), “run” (sound an alarm anyway before being discovered), and “fight back” (attempt to thwart the attacker’s activity).

Second, even without total access to the host and without an ability to damage the IDS itself, the intruder can influence the reporting channel. This is because a host-based system must communicate the alerts to the outside parties: email, syslog, SNMP, or other network connection are typically used. Some of these protocols can be disrupted even without having complete access to a host.

Third, host IDS usually presents a bigger administrative and management challenge than network intrusion detection. While it is sufficient to have one NIDS per network segment, HIDS should be installed on every monitored host. That complicates both system configuration and report/alert collection. In addition, having host IDS systems of different classes and from different vendors usually complicates alert correlation and aggregation. Security Information Management solutions can ease the load of analyzing host intrusion detection output.

Fourth, some of the HIDS classes will incur a performance penalty on the protected host. Some vendors report that their kernel-level HIDS incur an extra 5–10% CPU load. Integrity checking is extremely CPU and I/O heavy (if only during the periodic check) due to the underlying cryptographic algorithms.

It should be noted from our honeypot experience, that few of the less competent attackers will bother to check for the presence of the host IDS and/or use any of the above methods to disable it. While typical automated attack kits (autorooter + rootkit) do disable standard UNIX system logging and (sometimes) process accounting, they will not attempt to foil the operation of the integrity checker. One possible explanation is that people who fall victim to such an unsophisticated attack are usually not the users of system integrity checkers. However, in a recent case, the attacker did take steps that accidentally disabled the integrity checking: He simply “rm -rf /”ed the whole machine.

ATTACKS AGAINST INTEGRITY CHECKERS

Now let’s turn to integrity checkers (IC) and outline some attacks against them together with countermeasures. As was outlined above, the typical IC program computes the checksum and collects information about files (“initialize mode”). Then the program will periodically check for changes (using the “check mode”). In addition, the system admin can update the file signature after reconfiguring the system (“update mode”). Depending on the implementation of the IC program, each of those modes can be attacked.

ATTACKS AGAINST “CHECK MODE”

TROJAN BINARY

If an IC program uses a standard system binary to check the integrity (e.g., RPM’s /usr/bin/md5sum), one can replace the binary to report the right checksum for certain files. Using this simple method, the intruder can hide a small number of files from checksumming, but not from other checks (such as location). For the Linux RPM-based system case, replacing the RPM binary will work just as well.

COUNTERMEASURES

This attack is only provided as an example, since most IC programs use much more than just a checksum and often implement their own MD5 algorithm.

KERNEL-LEVEL MODULE (LINUX, SOLARIS, *BSD)

An attacker can deploy a malicious loadable kernel module (LKM) to remap the system calls. As a result the `open()` call to open a certain file for checking will be redirected to another target. Thus the original file moved to a different location by an attacker will be checksummed. Or, one can leave the `open()` call to open the original file left in place, but instead redirect the `execve()` to run the malicious program stored elsewhere (the approach used by `twhack.c` sample code in the “Bypassing Integrity Checking” article in *Phrack* #51). In addition, remapping some system library (`libc`) calls can accomplish the same task. The malicious system libraries for Linux and Solaris were observed in the recent system breaches.

COUNTERMEASURES

Implementing more checks will require the attacker to remap more and more system calls (which is a non-trivial programming challenge).

Example: Adore v.0.42 vs. AIDE and Tripwire

Adore LKM is a kernel-level backdoor for Linux and FreeBSD, featuring file, process, and connection hiding. Adore remaps `fork()`, `write()`, `open()`, `stat()` (=get file information), `close()`, `clone()` (=like `fork()`), `kill()`, `mkdir()`, and `getdents()` (=get directory entries)

system calls. By default, if you know the filename and location of a file, you will be able to look at it, but it will not show in the directory.

AIDE uses `open()` to query the files as shown in the above call trace excerpt (obtained by “`ltrace -S -f -r -C -s 1000 -o aide-trace aide -check`”):

```
22015    0.000249 SYS_open("/etc/security", 67584, 027777753350) = 5
22015    0.000238 SYS_open("/etc/smrsh", 67584, 027777753350) = 5
22015    0.000241 SYS_open("/etc/locale", 67584, 027777753350) = 5
```

Thus, if Adore is configured to hide the presence of a file, AIDE check will not report on the file. On the other hand, Tripwire will still catch the attacker, because it uses `read()` in its integrity checking after doing an `open()` based on its own records (and not the `getdents()` output which is also remapped):

```
22020    0.000075 SYS_read(3, "\177ELF\001\001\001", 1024) = 1024
22020    0.000075 SYS_read(3, "\177ELF\001\001\001", 1024) = 1024
```

It should be noted that by remapping more calls, even this can probably be circumvented.

FAKED REPORT

The attacker might be able to break the email sending functionality and then manually send a faked “All OK” report, modeled after the original report. Admittedly, this attack relies on the intruder’s ability to prevent the communication between integrity checking and the reporting station. However, it might require fewer privileges than the full-blown attack, such as “mail” group privileges vs. those of a “root” user. An even more malicious variant of this attack involves replacing the integrity checker binary with an “OK report generator” program, which sends the report to the sysadmin at specified intervals. This involves having root access, but can present a more permanent solution immune to any of the system changes.

If a different networked channel is utilized, it can also be attacked. Consider that having complete control of the target machine, the attacker might initiate any connection (even encrypted) or respond to any connection from an IDS management console. It should be noted that attacks of this kind have not been observed “in the wild.”

COUNTERMEASURES

Use of secure channels for reporting will stop this attack. Signed email will be considered secure only if someone actually checks the signatures on the reports. Otherwise, faking “signed” reports is just as easy as faking unsigned reports.

ATTACK BETWEEN CHECKS

While it sounds trivial, if an attacker manages to complete his activities between periodic checks and then restore the system to its original state, the IC program will not report an intrusion. It is impractical to expect hourly integrity checks for most environments (and that would cause heavy CPU utilization for cryptographic checksumming). However, if the original file is replaced, it will likely not be stored in the same disk location and the crime may be discovered.

COUNTERMEASURES

Daemon integrity checkers (such as the somewhat obscure “Samhain,” available at <http://www.la-samhna.de/samhain/>) do exist. Samhain is an impressive tool that boasts powerful defenses such as an encrypted and compressed executable binary, cryptogra-

phy support, steganographically shielded configuration files (can be merged with innocent GIF or JPEG images), a deceptive command line, and its own kernel-level hiding kit. These are in addition to secure reporting or information hiding in case the reporting channel is cut off.

As a side note, while some of the crypto-protocols used for integrity checking (such as MD5) were found to have collisions (i.e., different files having the same MD5 checksum), their impact on the security of real-world systems is minor, since it is likely not the weakest link for the typical IC deployment scenario. Still, better integrity checkers, such as Tripwire, use several different algorithms to eliminate this possibility.

ATTACKS AGAINST “UPDATE MODE”

For simple integrity checking programs, attackers will be able to run the “update mode” after modifying the system. Similarly, the attacker can modify the signature database directly. It will work only if the database is stored on the same system that is being checked (not a good idea).

COUNTERMEASURES

Update mode should require a password, and the database should be encrypted (done by some commercial integrity checkers) and, ideally, stored off-site on read-only media (in-depth defense). That will also help to prevent some insider and physical attacks.

ATTACKS AGAINST “INITIALIZE MODE”

The evident attack is that if the system is already trojaned, the initialize mode will create a set of signatures that establish the “compromised baseline.” The deployed trojans might also capture the password used for the database.

COUNTERMEASURES

Run the integrity checker right after installation and before connecting to the untrusted network. After creating the database, copy it and store it away from the protected computer.

Now let’s consider kernel-level solutions. We will briefly look at Linux StMichael loadable kernel module (LKM), designed to alert on the presence of malicious kernel modules (such as Adore). The module is also able to perform integrity checks on some of the data structures inside the kernel and to detect tampering with various kernel calls. StMichael also conceals itself (using the same tricks as adversaries such as Adore or knark).

For example, if StMichael is loaded and some other kernel module attempts to load in hidden mode and remap system calls, the malicious module will be revealed and (optionally) the changes to a kernel call table will be reversed with the log message:

```
Apr 24 18:32:13 anton kernel: 0(STMICHAEL)
:Kernel Structures Modified. Attempting to Restore.
```

In case the rootkit was loaded before StMichael, the module might be able to perform detection as well:

```
Apr 24 18:28:59 anton kernel: (STMICHAEL)
Possible LKM Rootkit Detected during Load.
```

Cryptographically signed system logs (while they sound attractive) do not measure up to a tried-and-true remote logging.

LOG DELETION/MODIFICATION

Now we are ready to briefly discuss log analyzers. Log analyzers are relatively easy to foil. Most attackers disable system logging and/or wipe system logs and process accounting records. Once root access is achieved, deleting or modifying system logs is easy. Numerous tools (e.g., clean, from THC toolkit, and others) exist to cleanly delete, log, and audit records from text and binary logs. Modern Linux/Solaris rootkits include such tools, and they are automatically activated upon rootkit installation, erasing all traces of log evidence. Now, if remote logging is enabled, the typical rootkit might alert the owner about the fact, but there is nothing it can do about it if the incriminating log messages were already shot across the network over UDP.

COUNTERMEASURES

Remote logging is the most commonly used and reliable measure against log tampering. Using a secure log server or a serial connection to drop off the logs is easy to implement and adds a lot to security.

Cryptographically signed system logs (while they sound attractive) do not measure up to a tried-and-true remote logging. However, it is well known that standard UNIX log transport uses the unreliable UDP protocol. Tools exist to flood the log server with messages and cause it to overflow, crash, or stop receiving messages. In addition, faked data injection can present another risk for some environments. Log processing and correlation engines can be made to reach the wrong conclusion if their correlation logic is known to the attacker. For example, sending faked messages seemingly from the FTP daemon might lead the log analysis program to believe that an FTP session has ended, while in fact the message was crafted by the attacker.

Moreover, a bug in a log monitoring program can lead to a compromise. A recent critical bug in LogWatch enabled attackers to gain local “root” privileges simply by crafting a simple shell script, which abuses temporary files created by LogWatch. More details on the exploit are available at <http://www.securiteam.com/exploits/5OP0S2A6KI.html>.

The obvious conclusion is that to prevent this and other vulnerabilities, log analysis IDS should be run on highly secured log aggregation servers and not on all production machines.

Conclusion

UNIX host security, while somewhat vaguely defined, contributes a lot to maintaining a secure computing environment. This paper introduced some of the issues that should be considered before the deployment of host-based intrusion detection. Let us also note that effective centralized reporting and audit trail analysis will significantly increase the value of host-based intrusion detection. It is crucial to have a central point for the HIDS to report to.

Another promising aspect of host security is application security. Ideally, host IDS should be able to understand the audit trails produced not only by system resources, but also by applications. In this case, a more comprehensive picture of security can be achieved.