Log analysis for intrusion detection.

by Daniel B. Cid (dcid @ ossec.net)

1 - Introduction

Log analysis is one of the most overlooked aspects of intrusion detection. Nowadays we see every desktop with an anti-virus, companies with multiple firewalls and even simple end-users buying the latest security related tools.

However, who is watching or monitoring all the information these tools generate? Or even worse, who is watching your web server, mail server or authentication logs? I'm not talking about pretty usage statistics of your web logs (like what webalizer does). I'm talking about the crucial security information that only few of these events have and nobody notices. A lot of attacks would not have happened (or would have been stopped much earlier) if administrators cared to monitor their logs.
We are not saying that log analysis is easy or that you should be manually looking at all your logs on a daily basis. Because of their complexity and generally high volume, automatic log analysis is essential. These are some of the things your analysis tool should do:

-Understand your logs. Know what is good and what is bad.
-Correlate the *bad* events looking for patterns that may indicate an attack or intrusion.
-Correlate the *good* events with the *bad* events (eg. multiple failed logins followed by a successful one).
-Correlate the *good* events (eg. too many sucessful logins for the same user across multiple hosts in a small period of time).

-Look for unusual patterns that are not in your good or bad list.

Of course that doing all of these steps is not easy. The main goal of this document is to show you how some threats can be detected by correlating specific patterns on your web, proxy and authentication logs. We are using OSSEC HIDS as an example because it not only does all the analysis we mention in here, but also has rules for multiple log formats, making our correlation simpler.

2 - Proxy log analysis.

When referring to proxy log analysis, we generally use squid as an example because it is the most used web proxy out there. When squid is implemented correctly, all traffic will pass through the proxy without no extra configuration on the user side. Because of that, we have full access to every page the users are acessing. Notice that what we detect on the proxy log analysis is generally of high importance, because they come from the inside or your network. The following are some of the issues we can find out by monitoring our proxy logs:

2.1 - Internal users scanning or attacking outside systems.

Basically, every time a user tries to access a non-existent page, squid will log the HTTP error code (generally 404 or 403). If you see multiple 400 error codes from the same source ip within a small period of time, a flag should be raised. If the user is acessing a page with broken links, it will cause a false positive, so we ignore the .gif, .jpg and .png extensions (among others). Using this approach, we can detect internal users trying to scan or gather information from outside systems or sites. A NIDS (Network-based Intrusion Detection System) or a signature-based analysis would miss it.

2.2 - Internal users with worms, trojans or virus.

A lot of worms have specific ways of accessing a web site or an external page. Detecting those accesses may indicate an internal infected user. For instance, in the case of the W32.Beagle worm, an infected machine would try to access pages with the "xxx3.php" or "blst.php" extension to notify the worm creator. If we see those entries in the logs, we know that something is wrong. Below is an OSSEC alert as an example of an infected user:

OSSEC HIDS Notification.
2006 May 11 11:00:00

Received From: (web-proxy) 192.168.2.1->/usr/local/squid/var/log/access.log
Rule: 5054 fired (level 12) -> "Infected machine with W32.Beagle.DP.'"
Portion of the log(s):

524 192.168.2.204 TCP_MISS/404 590 GET http://www.ordendeslichts.de/intern/xxx3.php? - DIRECT/81.201.107.6 text/html
3571 192.168.2.204 TCP_MISS/404 470 GET http://www.levada.ru/htmlarea/images/xxx3.php? - DIRECT/62.118.252.213
466 192.168.2.204 TCP_MISS/404 543 GET http://www.etype.hostingcity.net/mysql_admin_new/images/xxx3.php? - DIRECT/217.158.10.80 text/html
516 192.168.2.204 TCP_MISS/404 423 GET http//www.deadlygames.de/DG/BF/BF-Links/clans/xxx3.php? - DIRECT/81.169.145.95 text/html
528 192.168.2.204 TCP_MISS/404 423 GET http://stroyindustry.ru/service/construction/xxx3.php? - DIRECT/217.16.16.135 text/html
950 192.168.2.204 TCP_MISS/404 686 GET http://service6.valuehost.ru/images/xxx3.php? - DIRECT/217.112.42.95 text/html

505 192.168.2.204 TCP_MISS/404 1368 GET http://schiffsparty.de/bilder/uploads/xxx3.php? - DIRECT/212.227.94.133 text/html

2.3 - Invalid users in the network.

Some proxies require authentication from the user. In the case that the user does not have valid credentials, a log wil be generated. If we see one or two authentication errors, it may indicate that a user just forgot his or her password. However, seeing multiple errors from the same source ip, is a signal that something wrong is going on (specially if the attempts are for different user names). Look bellow at two examples of authentication error messages generated by squid:

1134746808.068 34 10.1.2.3 TCP_DENIED/407 2675 GET http://www.test.com/ user NONE/- text/html
1096907971.215 4 10.1.2.4 TCP_DENIED/407 3715 GET http://www.microsoft.com/isapi/redir.dll? - NONE/- text/html

2.4 - Proxy misuse or access violations.

If you setup a web proxy, you probably want it to be used to proxy only web traffic. However, some users may try to misuse some protocols similarities to send external e-mails or relay other protocols that otherwise would be blocked. For example, a commom Squid problem is users trying to use it to relay e-mails. It can easily be fixed by denying requests to port 25, but you still want to see who is trying to do that. By monitoring accesses to any unusual port, we can find the offender users. The following alert shows an internal user trying to relay e-mail:

OSSEC HIDS Notification.
2006 May 12 07:05:12

Received From: (web-proxy) 192.168.2.1->/usr/local/squid/var/logs/access.log
Rule: 5051 fired (level 10) -> "Multiple attempts to access forbidden file or directory from same source ip.'"
Portion of the log(s):

0 192.168.2.135 TCP_DENIED/403 1382 CONNECT 65.54.245.104:25 - NONE/- text/html
2 192.168.2.135 TCP_DENIED/403 1378 CONNECT 4.79.181.14:25 - NONE/- text/html
0 192.168.2.135 TCP_DENIED/403 1390 GET http://www.ebay.com/ - NONE/- text/html
3 192.168.2.135 TCP_DENIED/403 1378 CONNECT 4.79.181.14:25 - NONE/- text/html
5 192.168.2.135 TCP_DENIED/403 1392 GET http://www.yahoo.com/ - NONE/- text/html
6 192.168.2.135 TCP_DENIED/403 1384 CONNECT 66.135.192.123:80 - NONE/- text/html
2 192.168.2.135 TCP_DENIED/403 1380 CONNECT 66.94.230.75:80 - NONE/- text/html
420 192.168.2.135 TCP_DENIED/403 1390 GET http://www.ebay.com/ - NONE/- text/html
6 192.168.2.135 TCP_DENIED/403 1384 CONNECT 66.135.192.123:25 - NONE/- text/html

2.5 - Policy violations.

Some companies do not allow access to external webmails or pornographic web sites at work. With squid, you can directly block those accesses, but you need to know who is trying to access the prohibited pages. Using log analysis, you can set up a list of undesired web sites or IPs to alert or block on them. By default this is not enabled on OSSEC because each company has its own policies, but we have a long list of common webmail sites, pornographic IPs/sites, gaming sites, etc.

3- Web log analysis.

Some people trust a NIDS (Network-based IDS), like Snort, to detect attacks against their web applications. However, most NIDS do not have a good correlation engine for web traffic and they miss a lot of important information, including internal errors, return codes, etc. If your site runs on SSL, a NIDS is completely useless. These are some of the issues we can detect by monitoring our web logs:

3.1 - Web scan or information gathering.

When someone is trying to get illegal access to your systems, he or she will probably scan them looking for vulnerable applications (like old awstats or phpbb versions). That will cause your web server to generate many 400 error messages. If we detect a few of them within a small period of time from the same source ip, we can raise the flag. Sometimes this can generate false positives if your site has broken links, so we ignore .gif, .jpg and .png extensions (like we do on squid logs). Using this kind of correlation, we can detect new worms or zero day vulnerabilities in web apps. The following multiple 404 errors (looking for xmlrpc) are examples of web scan:

200.179.157.1 - - [13/Jan/2006:01:03:30 -0200] "POST /blog/xmlrpc.php HTTP/1.0" 404 288
200.179.157.1 - - [13/Jan/2006:01:03:31 -0200] "POST /blog/xmlsrv/xmlrpc.php HTTP/1.0" 404 295
200.179.157.1 - - [13/Jan/2006:01:03:32 -0200] "POST /blogs/xmlsrv/xmlrpc.php HTTP/1.0" 404 296
200.179.157.1 - - [13/Jan/2006:01:03:33 -0200] "POST /drupal/xmlrpc.php HTTP/1.0" 404 290
200.179.157.1 - - [13/Jan/2006:01:03:35 -0200] "POST /phpgroupware/xmlrpc.php HTTP/1.0" 404 296
200.179.157.1 - - [13/Jan/2006:01:03:36 -0200] "POST /wordpress/xmlrpc.php HTTP/1.0" 404 293
200.179.157.1 - - [13/Jan/2006:01:03:44 -0200] "POST /xmlrpc/xmlrpc.php HTTP/1.0" 404 290
200.179.157.1 - - [13/Jan/2006:01:03:46 -0200] "POST /xmlsrv/xmlrpc.php HTTP/1.o

3.2 - Successful and failed attacks against web applications.

NIDS only look at some specific content before they alert that an attack is happening. However, using log analysis, we can see whether the attack worked or not. Even better, we can see the logs of SSL connections that a NIDS would not be able to see. With a small set of rules, we can detect SQL injections, directory transversal problems, attempts to execute commands and multiple other attacks (knowing for sure if they sucessed or not).

For example, to detect SQL injections, we look for some SQL commands like **select**, **where** or **from**. The same applies to detect attempts to execute commands. Every system has a sub set of commands that we need to monitor, like cat, grep, wget, dir, ls, etc. We also look for spaces, new lines or null terminators, because they are widely used (and necessary) on most attempts to execute commands. OSSEC rules for web logs contain more information.

For example, on these two awstats attacks. We see some common system commands, we see separators and some unusual characters in the url. Looking at the HTTP result code, we know that one was sucessfull and the other was not (error 404 and 200). With access to this information we can increase the severity of the one that worked and minimize the severity on the one that failed. On OSSEC, we do this for the **common web attacks**. If they are successful, we increase their severity and immediatly alert the admin and execute an active-response.

a.b.c.d - - [13/Jan/2006:01:07:21 -0200] "GET /awstats/awstats.pl?configdir=|
echo;echo%20YYY;cd%20%2ftmp%3bwget...;echo%20YYY;echo|HTTP/1.0" 404 291
a.b.c.d - - [14/Jan/2006:01:01:25 -0200] "GET /cgi-bin/awstats.pl?configdir=|

echo;echo%20YYY;cd%20%2ftmp%3bwget...;echo%20YYY;echo\HTTP/1.0" 200 291

In addition to that, the following windows and mrgt attacks show some other examples of what to look for:

a.b.c.d - - [12/Apr/2006:08:05:46 -0300] "GET /rpc/..%%35%63..%%35%63..%%35%63..%%35%63/winnt/system32/cmd.exe?/c+dir+c:\\+/OG HTTP/1.0" 400 294 200.179.154.180 - - [12/Apr/2006:08:05:47 -0300] "GET /cgi-bin/%2E%2E%2F%2E%2E%2F%2E%2E%4E%4E%54%2F%73%79%73%74%65%6D%33%32%2Fping.exe%20127.0.0.1
200.255.5.3 - - [12/Apr/2006:08:05:43 -0300] "GET /cgi-bin/mrtg.cgi?cfg=/../../../../../../etc/passwd HTTP/1.0" 404 289
200.255.5.3 - - [12/Apr/2006:08:05:43 -0300] "GET /cgi-bin/mrtg.cgi?cfg=/../../../../../../winnt/win.ini HTTP/1.0" 404 289

3.3 - Web server problems.

A lot of problems on your web server can be detected by looking at its logs. For example, the following errors would go unnoticed if we weren't monitoring it (OSSEC alert sent by e-mail):

OSSEC HIDS Notification.
2006 May 12 04:40:17

Received From: (web-server) 10.1.1.25->/var/log/apache/error_log
Rule: 102 fired (level 7) -> "Unknown problem somewhere in the system.'"
Portion of the log(s):

**** glibc detected *** corrupted double-linked list: 0xb7daca0c ****

OSSEC HIDS Notification.
2006 May 10 16:41:31

Received From: (intra-server) 10.1.2.41->/var/log/apache/error_log
Rule: 102 fired (level 7) -> "Unknown problem somewhere in the system.'"
Portion of the log(s):

[client 201.25.30.140] PHP Fatal error: Allowed memory size of 31457280 bytes exhausted (tried to allocate 39518206 bytes) in /home/site/htdocs/components/com_search/search.php on line 172, referer: http://www.mysite.com.br/index.php?option=Itemid=5&searchword=+SNORT+%2B+MYSQL+%2B+APA

4- Auth log analysis.

Analysing authentication messages are extremally crucial. First, you can see who accessed what and when specifically. Second, you can see if someone is acessing something he (she) shouldn't be. You can also find out internal misuse by seeing the hours and systems users are trying to access. In addition to that, brute force attacks and other password guessing problems can be detected.

4.1 - Users acessing what they should not be.

Most log analysis tools only alert on failed logins attempts. However, what happens if a valid user is acessing a device he (she) is not supposed to be? Or acessing when he (she) shouldn't be at work? During analysis, we need to create a baseline of all users and the systems they are allowed to access. This technique is used by OSSEC and called FTS (First Time Seen). Every time a user access a system that OSSEC has never seen this user acessing before, it will generate an alert. For the first days, OSSEC will be *"learning"* which users can acess which systems, causing some extra alerts. However, after a while, a baseline will be created and only deviating accesses will be alerted. With FTS, you can discover invalid users in the network or users acessing what they shouldn't be.

4.2 - Login from a system user.

System users are used for internal purposes only and we should never see a login for them. If we see any of them acessing a machine through ssh, telnet ,ftp or any other method, we need to raise a flag. It may indicate that an application or process was compromised. OSSEC has a list of system users (like apache, mysql, nobody, portmap, www, bin, etc) that is very helpful in identifying them. Example of OSSEC alert when user **nobody** logged in:

OSSEC HIDS Notification.
2006 May 12 08:59:45

Received From: (auth1) 192.168.20.55->/var/log/messages
Rule: 1601 fired (level 12) -> "System user sucessfully logged on the system."'
Portion of the log(s):

sshd[23410]: Accepted password for nobody from 10.1.2.3 port 42802 ssh2

4.3 - Multiple failed logins.

Brute force and dictonary attacks are becoming more and more common but you can block them by monitoring your authentication logs . First, if you see many failed logins from the same source IP within a few minutes, it is probably an attack. Second, if you see multiple failed logins from different IPs, it is probably an distributed attack or something is wrong internally with your system. Bellow is an OSSEC alert for a SSHD brute force attack:

OSSEC HIDS Notification.
2006 May 11 21:17:07

Received From: /var/log/messages

Rule: 1512 fired (level 10) -> "SSHD brute force trying to get access to the system.'"
Portion of the log(s):

sshd[9370]: Failed password for invalid user admin from 200.30.175.162 port 58257 ssh2
sshd[9370]: Invalid user admin from 200.30.175.162
sshd[9368]: Failed password for invalid user fluffy from 200.30.175.162 port 58212 ssh2
sshd[9368]: Invalid user fluffy from 200.30.175.162
sshd[9366]: Failed password for invalid user slasher from 200.30.175.162 port 58109 ssh2
sshd[9366]: Invalid user slasher from 200.30.175.162
sshd[9364]: Failed password for invalid user sifak from 200.30.175.162 port 58030 ssh2

4.4 - Multiple failed logins followed by a successful login.

This one is a serious event. If you see attempts for multiple users and multiple passwords from the same source IP, followed by a success, the attacker probably got luck. It may be a valid user that forgot the password and after many errors, got it right, causing a false positive. However, if multiple users were attempted, the false positive rate goes down.

5 - Conclusion

This document is not finished yet. We plan to add more information regarding mail, NIDS, Firewall, generic syslog and windows log analysis. We are also accepting more ideas and contributions to make it more reliable and useful to the security community.