

Linux Programming Activities

Contents

Linux Programming Activities	1
Activity 7-1: Creating an HTML Web Page	1
Activity 7-2: Writing a Hello World Perl Script	2
Activity 7-3: Writing a Perl Script Using Ping and Notify-Send	3
Activity 7-4: Writing a Cross Platform Python Script for Security Testing	5
Assignment Submission.....	7

Time required: 90 minutes

NOTE: Please read the book as you are going through these activities to gain an understanding of programming concepts.

Activity 7-1: Creating an HTML Web Page

Objective: Create an HTML Web page.

Description: As a security tester, you might be required to view Web pages to check for possible Web security issues. A basic knowledge of HTML can help you with this task. In this activity, you create a simple HTML Web page and then view it in your Web browser.

1. Start your computer in **Windows**.
2. You can use Notepad, Visual Studio Code, or any text editor to create an html file named: **myweb.html**
3. In the new document, type the following lines, pressing Enter after each line:

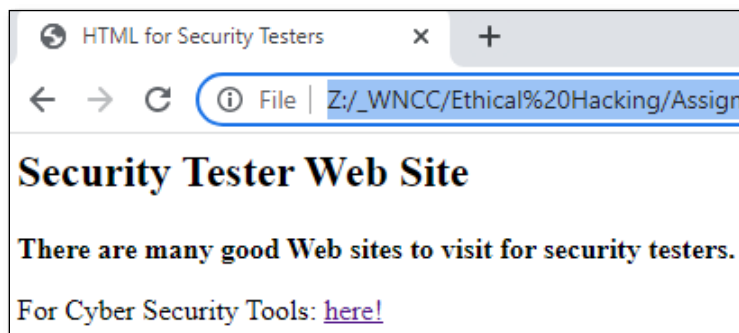
```

1  <!-- A practice HTML page -->
2  <html>
3
4  <head>
5  |   <title>HTML for Security Testers</title>
6  </head>
7
8  <body>
9  |   <h2>Security Tester Web Site</h2>
10 |   <p>
11 |       <b>There are many good Web sites to visit for security testers. </b>
12 |   </p>
13 |   <p> For Cyber Security Tools:
14 |       <a href="https://www.sans.org/tools/?msc=main-nav">here!</a>
15 |   </p>
16 |
17 </body>
18 </html>

```

4. To test whether you have created the Web page correctly, start File Explorer and navigate to the location of your file. Double Click **myweb.html**
5. If you entered the information correctly, your Web page should look like the example run.
6. Click the **here!** hyperlink you created to check whether you're sent to the correct Web site. If not, make corrections to your HTML code.

Example run:



Activity 7-2: Writing a Hello World Perl Script

Objective: Write a Perl script using **geany**

Description: Security professionals and hackers alike use the Perl scripting language. Many hacking programs are written in Perl, so any skills you develop in this language will help you in your career.

In previous activities, you used nano as a text editor in Kali Linux. In this activity, you write a basic Perl script using the graphical programming environment called **geany**. Unlike nano, **geany** is a GUI program that lets you click to navigate instead of relying on keyboard commands.

1. Boot your computer into **Kali Linux**.
2. Open a terminal window.
3. Type **geany first.pl** and press Enter.
(If geany is not installed: **sudo apt install geany**)
4. Enter the following code:

```
1 # I should always have documentation/comments in my scripts!
2
3 # This code displays "Hello security testers" to the screen
4
5 print "Hello security testers!\n\n";
```

5. Open a new terminal prompt:

```
(user@kali)-[~]
$ cd ~/Desktop

(user@kali)-[~/Desktop]
$ perl first.pl
Hello security testers!
```

6. If your code does not contain errors, your screen should look like the example run.
If you receive error messages, read through your file and compare it with the lines of code in this activity's steps. Correct any errors and save the file again.

Activity 7-3: Writing a Perl Script Using Ping and Notify-Send

Objective: Write a Perl script that uses branching, looping, and testing components.

Description: Security professionals often need to automate or create tools to help them conduct security tests. In this activity, you write a Perl script that uses the notify-send command and a for loop to select IP numbers from the classroom range your instructor has provided.

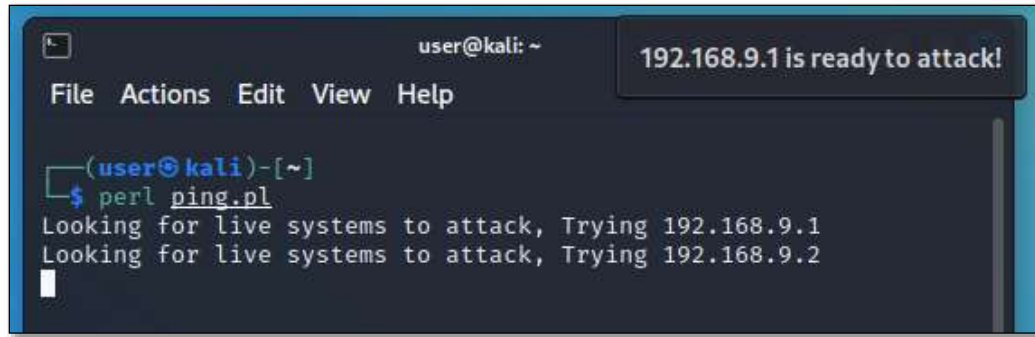
You can use the following reference for the Perl ping command:

<http://perldoc.perl.org/Net/Ping.html>.

1. Boot your computer into **Kali Linux**.
2. Open a terminal window.
3. Type **geany ping.pl** Press Enter.
4. Enter the following code.

```
1  # Filename: ping.pl
2  # Program to ping workstations in the local network
3  # If the ping is successful, a message is sent to the screen
4  # Program assumes a Class C address (w.x.y.z)
5  # where w.x.y is the network portion of the IP address
6  # The "z" octet will be incremented from 1 to 254 with a for loop
7
8  # Load the Net library
9  use Net::Ping;
10
11 # Create new ping object with default settings
12 $p = Net::Ping->new();
13
14 # Initializes the class_IP variable to hold your network ID
15 # Change to reflect your topology
16 $class_IP = "192.168.9";
17
18 # The for loop, which increments the last octet
19 # of the network IP address to all available IP addresses in your class
20 for ($z=1; $z<255; $z++) {
21     # Creates the host to be scanned for this iteration of the for loop
22     $wkstation = "$class_IP.$z";
23
24     # Displays status message
25     print "Looking for live systems to attack, Trying $wkstation \n";
26
27     # Sends notification message to screen if ping is successful
28     system("notify-send '$wkstation is ready to attack!'") if $p->ping($wkstation);
29 }
```

Example run:

A screenshot of a Kali Linux terminal window. The window has a title bar with a close button and the text 'user@kali: ~'. Below the title bar is a menu bar with 'File', 'Actions', 'Edit', 'View', and 'Help'. A notification bubble in the top right corner says '192.168.9.1 is ready to attack!'. The terminal prompt is '(user@kali)-[~]'. The user has entered the command '\$ perl ping.pl'. The output shows two lines: 'Looking for live systems to attack, Trying 192.168.9.1' and 'Looking for live systems to attack, Trying 192.168.9.2'. A cursor is visible on the line following the second output line.

```
user@kali: ~
File Actions Edit View Help
192.168.9.1 is ready to attack!
(user@kali)-[~]
$ perl ping.pl
Looking for live systems to attack, Trying 192.168.9.1
Looking for live systems to attack, Trying 192.168.9.2
```

Activity 7-4: Writing a Cross Platform Python Script for Security Testing

Objective: Write a cross platform Python script that uses branching, looping, and testing components.

Description: We have already worked with Python extensively in this class. We are going to jump right in without a hello world program.

Security professionals often need to automate or create tools to help them conduct security tests. In this activity, you write a Python script that uses the ping command and a for loop to ping IP numbers for an entire class C network.

Use the class C network of your home network. If a ping is successful, it indicates that a computing device has been found at that IP address.

1. Start your computer in **Windows**.
2. Use the network address of your local network.
3. Create a new file called **pythonping_scanner.py**.
4. Enter the following code.

```

1  # Filename: pythonping_scanner.py
2  # This program prompts the user to enter a class C network address
3  # it uses the pythonping library to detect active devices
4  # at each possible IP address
5
6  # Windows: pip install pythonping
7  # Linux: sudo apt install python3-pythonping
8  # In Linux: Use sudo to run script
9  from pythonping import ping
10
11 print("\n ***** Python Network Ping Scanner *****")
12 print("Press CTRL C to exit")
13 # Prompt the user to input a network address and press Enter.
14 net_addr = input("Enter a class C network address (ex.192.168.1): ")
15
16 # n will range from 0 to 254 which
17 # the program will use to create all IP addresses
18 for n in range(254):
19
20     # Create the next IP address to ping
21     # by combining net_addr with the current value of n
22     current_target_ip = f"{net_addr}.{n+1}"
23
24     result = ping(
25         current_target_ip,
26         count=1,          # Number of pings
27         timeout=1         # Timeout in seconds
28     )
29
30     if result.rtt_avg_ms < 300:
31         # If less than 300ms, there is a target
32         print(current_target_ip, "Active - ping responded: device detected")
33     else:
34         # If greater than 300ms, no response
35         print(current_target_ip, "Inactive - no ping response")

```

5. Open a Windows command prompt and navigate to the folder containing your **pythonping_scanner.py** file.
6. In Windows: Run your script by typing **python pythonping_scanner.py** and pressing Enter.
7. If you have no errors, your program should begin pinging IP addresses, as shown below.

8. You can let the program run to completion or, to terminate the Python script, press **CTRL C**

Example run in Windows:

```
***** Python Network Ping Scanner *****
Press CTRL C to exit
Enter a class C network address (ex.192.168.1): 192.168.9
192.168.9.1 Active - ping responded: device detected
192.168.9.2 Inactive - no ping response
192.168.9.3 Inactive - no ping response
192.168.9.4 Inactive - no ping response
192.168.9.5 Inactive - no ping response
192.168.9.6 Inactive - no ping response
192.168.9.7 Inactive - no ping response
192.168.9.8 Inactive - no ping response
192.168.9.9 Inactive - no ping response
192.168.9.10 Active - ping responded: device detected
192.168.9.11 Inactive - no ping response
```

This is a cross platform script, copy and paste the code into Linux. Follow the directions in the program to install the **pythonping** library in Linux.

sudo python3 pythonping_scanner.py

Example run in Linux:

```
(user@kali)-[~]
$ sudo python3 pythonping_scanner.py
[sudo] password for user:

***** Python Network Ping Scanner *****
Press CTRL C to exit
Enter a class C network address (ex.192.168.1): 192.168.9
192.168.9.1 Active - ping responded: device detected
192.168.9.2 Inactive - no ping response
192.168.9.3 Inactive - no ping response
192.168.9.4 Inactive - no ping response
192.168.9.5 Inactive - no ping response
192.168.9.6 Inactive - no ping response
192.168.9.7 Inactive - no ping response
192.168.9.8 Inactive - no ping response
192.168.9.9 Inactive - no ping response
192.168.9.10 Active - ping responded: device detected
192.168.9.11 Inactive - no ping response
192.168.9.12 Inactive - no ping response
```

Assignment Submission

1. Attach all program files.
2. Attach a screenshot of each successful program run.

3. Submit the assignment in Blackboard.