CO2515 Coding for Penetration Testing Worksheet 2

Aim – To introduce students to some basic PT coding using Python3 and Scapy with IPv6. Only basic programming skills are assumed. All code is written to be as simple as possible.

NOTE – The techniques discussed in this worksheet should only be used on networks with the full written permission of the network owner otherwise you may be in breach of the GDPR.

|  |  |
| --- | --- |
| 1. | In this Lab you will be able to create tools which can test the operating system of a target and to be able to create a simple port scanner. |
| 2. | Create a new file called testOS.py and set it to run Python script and a description of the script. |
| 3. | **#!/usr/bin/python3**  **"""**  **This module sends detects the operating system of a target.**  **"""**  **from scapy.all import \*** |
| 4. | Now we need to send and receive a packet to a target we can do this with the **sr1()** function which is a variant of **sr()** but only sends one packet. Add this line to the bottom of your code: |
| 5. | **pkt= sr1(IPv6(dst="fd80:1234:5678:2::254")/ICMPv6EchoRequest())** |
| 6. | To save the inconvenience of adding the destination address this time you will be introduced to the **input()** function. This is simply a way of prompting the user to input text when running a piece of code.  Add this to the piece of code you created and change the destination address to the variable which stores the input. |
| 7. | **target = input("Input target IPv6 address? : ")** #Prompt target IPv6 Address  **pkt= sr1(IPv6(dst=target)/ICMPv6EchoRequest())** |
| 8. | The code currently just sends and receives a packet. To check the operating system an if() statement is required to check to hop limit of the received packet. Add the following underneath the rest of the code: |
| 9. | **if pkt.hlim < 65:** #If hop limit is less than 65 than the operating system is Linux,  #if not then it is windows.  **os = "Linux"**  **else:**  **os = "Windows"**  **print("Operating system is: " + os)** |
| 10. | Hop-limit **(hlim)** in IPv6 is the same as time-to-live **(ttl)** in IPv4. Linux machines typically have a default hop limit/ttl value of 64 whereas Windows machines have a value of 128 although this could be changed manually. The code simply checks if the hop limit value matches with a Linux or a Windows Operating system and displays it.  NOTE: The previous script can generally be used to determine which Operating system the target is since Windows and Linux are common. However, many more operating systems exist with a different ttl/hlim. |
| 11. | **#!/usr/bin/python3**  **"""**  **This module sends detects the operating system of a target.**  **"""**  **from scapy.all import \***  **target = input("Input target IPv6 address? : ")**  **pkt= sr1(IPv6(dst=target)/ICMPv6EchoRequest())**  **if pkt.hlim < 65:**  **os = "Linux"**  **else:**  **os = "Windows"**  **print("Operating system is: " + os)** |
| 12 | Use: **chmod a+x testOS.py**  And run the code: **sudo ./testOS.py** |
| 13 | **Building a port-scanner**  For the port scanner we will be using a TCP SYN scan. This involves sending a TCP SYN packet to a destination port on the target. The attacker initiates a 3-way communication and then waits for a response from the server. If the server responds with a SYN/ACK it means the port is open and the attacker then sends an RST (reset). Whereas, if the server responds with an RST itself it means the port is closed. |
|  |  |
| 14. | Create a new file called **portscanner.py** and send and receive a TCP packet to the target http port 80 from source port 333. |
|  | **#!/usr/bin/python3**  **“””**  **This module is a basic port scanner**  **created using Scapy**  **“””**  **from scapy.all import \***  **target = input("Input target IPv6 address? : ")**  **pkts = sr(IPv6(dst=target)/TCP(sport=333, dport = [21,80]))** # ’sr’ sends and receives a packet, ‘sport’ and’ dport’ set the source and destination ports |
| 15. | To see the results, add this beneath the code: |
| 16. | **ans,unans = pkts** # sr returns both answered and unanswered packets, this command selects the set stored in pkts  **ans.summary()** # the summary of the answered packets shows which ports respond with RA (closed) and which with SA (opened) |
| 17. | This should be the result:  IPv6 / TCP fd80:1234:5678:2::100:333 > fd80:1234:5678:2::254:ftp S ==> IPv6 / TCP fd80:1234:5678:2::254:ftp > fd80:1234:5678:2::100:333 RA  IPv6 / TCP fd80:1234:5678:2::100:333 > fd80:1234:5678:2::254:http S ==> IPv6 / TCP fd80:1234:5678:2::254:http > fd80:1234:5678:2::100:333 SA  NOTE: If the script fails the handshake check if the firewall is disabled on the target: **sudo ufw disable** |
| 18. | Remove the last line statements just added and replace them with this: |
| 19. | **for snd,rcv in ans:**  **if rcv.sprintf(r"%TCP.flags%")=="SA":**  **print (rcv.sprintf(r"%TCP.sport%" "OPEN"))** |
| 20. | Run the code and see the results. |
| 21. | Now we can create code which gives a range of port numbers which can be sent using the scapy code enter use the following code under the piece just added. This requests the user to enter the number of the first and last port they want to scan. The **range** function is then used to store all the numbers between the first and last ports which will be held in **dstlist**. The **+1** is required to compensate as number start with 0 in Python.  **fport = int(input("Enter firstport range: "))** #Enter the first and last port number. The range between the two fields will be stored in dstlist  **lport = int(input("Enter lastport: "))**  **lport +1**  **dstlist = list(range(fport, lport))**  #Add the dstlist variable to the dport field. |
| 22. | So far, your code should look like this: |
| 23. | **#!/usr/bin/python3**  **from scapy.all import\***  **"""This is a basic portscanner"""**  **target = input("Input target IPv6 address? : ")**  **fport = int(input("Enter firstport range: "))**  **lport = int(input("Enter lastport: "))**  **lport +1**  **dstlist = list(range(fport, lport))**  **pkt=sr(IPv6(src="fd80:1234:5678:2::100", dst=target)/TCP(sport=333, dport=dstlist, flags="S"))**  **ans, unans = pkt**  **for snd,rcv in ans:**  **if rcv.sprintf(r"%TCP.flags%")=="SA":**  **print (rcv.sprintf(r"%TCP.sport%" "OPEN"))** |
| 24. | From the terminal use **chmod a+x** portscanner.py  Run the script and examine the result. |
| 25. | Try changing the output to show other information such as also displaying the closed ports with the RA flags and an **elif** statement. |