Scripting for Cybersecurity
Assignment 2

Attack Automation

Assignment Summary

The net_attack.py script will automate the process of discovering weak usernames and passwords being used for services running on a host. The script will read a file containing a list of IP addresses. For each IP address in the list the script will scan the ports on that host, and attempt to bruteforce the login for detected services.

The script will take in the following parameters:

```
-t  -> Filename for a file containing a list of IP addresses
-p -> Ports to scan on the target host
-u -> A username
-f -> Filename for a file containing a list of passwords
```

Example usage would look like this:

```
./net_attack.py -t my_ip_list.txt -p 22,23,25,80 -u admin -f my_password_list.txt

./net_attack.py -t ip_list.txt -p 22 -u root -f passwords.txt
```

Files containing IP addresses and passwords can be found on Canvas.

Part 1 – Set-up (20%)

Create a script called “net_attack.py”.

Add a main function which contains code to read the arguments passed in by the user from the command line when the script was launched.

Add a function help(), which will print out the usage of the tool. If the user launches the script without the necessary arguments, this function should be called and should print out the arguments which a user can include when running the script as well as example usage.

Part 2 – Read IP addresses (10%)

Create a function “read_ip_list(ip_file)”. This function takes in 1 parameter, ip_file, which will be the name of the IP address file provided by the user through the -t option.

The function should open the file and read the contents of the file into a list, where each element in the list is an IP address. The function should return this list.

In your main function, after you have read in the arguments provided by the user, call the read_ip_list function and store the returned list in a variable. This variable now holds the list of IP addresses.

Part 3 – Verify connectivity (10%)

Create a new function “is_reachable(ip)”. This function will take in an IP address as a parameter.

This function will be used to check connectivity with a given IP address.

In this function use Scapy to send an ICMP request to the IP address. If a reply is received then the function should return True. If no response is received then the function should return False.

In your main function, for each IP address in your IP addresses list, call the is_reachable function to test connectivity with that IP address. If the is_reachable function returns False then remove that IP address from the IP address list. If the is_reachable function returns True then the IP address should stay in the list.

The list of IP addresses should now only contain IP addresses which can be pinged.

Part 4 – Port scan (15%)

Create a function “scan_port(ip, port)”. This function takes in an IP address and port number as parameters. The function will be used to scan the given port on the given IP address to check if it’s open. It will use a SYN scan to scan the port.

In the function, use Scapy to create an IP packet with a destination address of the given IP. Use Scapy to then create a TCP header containing the destination port of the given port, and set the TCP flags to be the appropriate flags for a SYN scan.

The function should then construct and send the packet to the given IP address. If a reply is received determine whether the port is open or closed based on the reply. If no reply is received then you can assume that the port is closed. The function should return True if the port is open and False if the port is closed.

In your main function, parse the port numbers provided by the user into a list. For each port number in the list call the scan_port function. If the function returns True then print some text to indicate that the port is open. If the function returns False then print some text to indicate that the port is closed.

Part 5 – Bruteforce Telnet (15%)

Create a function “bruteforce_telnet(ip, port, username, password_list_filename)”. This function takes in an IP address, a port number, a username, and a filename. The username should be the username provided by the user through the -u option. The filename should be the filename of the password list provided by the user through the -f option. This function will be used to attempt to bruteforce the Telnet username and password for a given host.

In the function, first open the file containing the password list. Next, for each password in the file, attempt to create a Telnet connection with the target host using the given IP address and Port number. Use Python’s Telnetlib to establish the Telnet connection. Attempt to log into the Telnet service with the given username and a password from the file. Detect whether or not the username and password combination has worked. Continue this operation until a working username and password combination has been found or you have run out of passwords to try.

If a working username and password combination has been detected then the function should return a string the username and password in a format like this “username:password”.

If no working username and password combination was found then return an empty string.

In your main function, when performing port scanning, if port 23 is found to be open then call the bruteforce_telnet function. If a working username and password combination has been found then print a message to alert the user and print the working combination.

Part 6 – Bruteforce SSH (10%)

Create a function “bruteforce_ssh(ip, port, username, password_list_filename)”. This function should perform a similar operation to the function created in part 5.

Instead of creating a Telnet connection, this function should attempt to create an SSH connection. Use the Python Paramiko library to create the SSH connections. As with the function in part 5, if a working username and password combination has been detected then return a string containing that combination. If no working combination was found then return and empty string.

In your main function, when performing port scanning, if port 22 is found to be open then call the bruteforce_ssh function. If a working username and password combination has been found then print a message to alert the user and print the working combination.

Part 7 – Bruteforce web login (5%)

Create a function “bruteforce_web(ip, port, username, password_list_filename)”. This function should perform a similar function to the functions from parts 5 and 6. Rather than Telnet or SSH this function will bruteforce the login credentials for a web application.

This function should use the Python requests library to send a HTTP GET request to the given IP address and given port number. It should detect whether a webpage was returned on not. If a web page was detected then the function should try to access http://ip_address:port_number/login.php

If a webpage is detected at that URL then the function should attempt to bruteforce the username and password for the site using the given username and passwords in the password list file. You can assume that the HTTP POST parameters are “username” and “password”. There data posted would therefore be like ‘{“username”: “someuser”, “password”: “somepassword”}’.

In your main function, when performing port scanning, if ports 80, 8080, or 8888 are found to be open then call the bruteforce_web function. If a working username and password combination has been found then print a message to alert the user and print the working combination.

Part 8 – Deploying files (5%)

Modify your code to accept a new command line parameter:

-d -> File to deploy on target machine

It would be used like:

```
./net_attack.py -t ip_list.txt -p 22 -u root -f passwords.txt -d test.txt
```

Modify your code so that when a working username and password combination is found for Telnet or SSH your script will connect to the target host and run commands that will cause the file specified by the user with the -d option to be transferred to the target host.

How you transfer the file to the target host is up to you. The only requirement is that it is transferred over the network.

Part 9 – Self-Propagation (10%)

Modify your code to accept a new command line parameter:

```
-L -> Local scan
-P -> Propagate
```

It would be used like:

```
./net_attack.py -L -p 22,23 -u root -f passwords.txt -P
```

Modify your code so that instead of a list of IP addresses, the user can use the -L option to have the scan performed on local networks. To do this, your script should get a list of the network interfaces on the device, get the IP address for each interface, and use the retrieved IP address to generate a list of addresses for the network. You may assume the network is a /24 network (i.e. only the last octet changes). Your script should then scan each of the IP addresses in your generated list.

The -P option will cause both the net_attack.py script and the passwords.txt files to be copied to the target host.

When a working username and password combination is found for Telnet or SSH your script will

- Connect to the Telnet or SSH server
- Detect whether this script is already on the server. If it is then skip to next target
- Deploy itself to the target server
- Deploy the list of passwords to the target server
- Run the net_attack.py script on the target with the -L and -P options

To test this, try targeting one IP address initially from the attacker, and use the Mininet terminal windows to observe whether the script is being propagated to other hosts in the Mininet network. Copy the files to the server_x directory on the target, where x is the server number.

Submission Details

This assignment is due on Friday 16th of December 2022. Submit your Python file to Canvas.

Your own original work should be submitted to Canvas. **Your code should include comments which explain how your code works.**

Submissions up to 1 week late will be subject to a 10% penalty.
Submissions up to 2 weeks late will be subject to a 20% penalty.
Submissions more than 2 weeks late will be subject to a 100% penalty.

The CIT late submission policy and the CIT plagiarism policy can be found on the CIT website.