# Homework 10 - Pivoting

## Summary

You have successfully gained access to one machine on a target corporate network, BockCorp. BockCorp doesn't have the best security policies and thinks their liberal use of network segmentation will protect them in the event of a breach. You won't have much time during the actual engagement on their live network, so your task is to write a program ahead of time that uses SSH port forwarding to navigate throughout BockCorp's multiple networks to compromise as many machines as possible. Your program's goal is twofold: compromise as many machines as possible, and find a specific file **flag.txt**, located in the root ("/") of some computer in their network. We do not know any credentials on the network, so as your worm navigates the corporate network, it will need to collect SSH keys and crack passwords in order to continue pivoting.

## Requirements

You should submit a python executable named `implant.py`.

Your program should print all credentials it collects throughout BockCorp's corporate network. It will navigate the network by logging in over SSH and setting up port forwarding as needed to continue traversing the networks. Your worm should be able to handle an arbitrary network layout and an arbitrary depth of networks away from your starting machine. Your program will collect credentials as it goes in two ways: SSH keys and cracking passwords.

Cracking passwords: On machines in which you have root credentials, you should get the /etc/shadow file and attempt to crack the passwords for each of the accounts in it. All of the passwords will be towards the top of the rockyou.txt list, and will not take long to crack.
SSH Keys: *(Optional - Extra Credit)* On all machines you get access to, your program should search for SSH keys in the home directory of the user's account you compromised.

As you encounter new machines, you should try all of the credentials you have collected so far. Note that you will likely not gain immediate access to all of the machines you encounter - credentials discovered later on during your network traversal may help give

you access to machines you've seen previously. As you collect these credentials, you should print them out on the screen to the user.

Your program should be written in Python and be entirely self-contained. You are welcome to use any python libraries that come pre-installed on Cypherpath's Kali Linux image.

## Description

Each computer in the BockCorp network contains a file named **servers.txt** located in every user's home directory. The **servers** file will contain any number of newline-delimited IP:Port combinations. For example, a servers file could have the following contents:

10.2.0.4:22
10.2.0.9:22
10.3.0.44:43334

Each of these lines corresponds to a valid, listening SSH server. You must attempt to log in to all machines in the target file, set up a port forwarder, and continue execution until the entire network has been compromised.

## Recommendations

As you traverse the network, you will effectively perform a breadth first search on the network. To make this easier, we recommend maintaining a queue (an array) of target IP addresses of machines you have encountered. If you cannot log into an IP with the credentials you currently have, simply put it back on top of the queue to revisit when more credentials have been obtained.

In developing your program, paramiko will be very useful to you to manage SSH sessions. Additionally, paramiko has an implementation of a port forwarder at https://github.com/paramiko/paramiko/blob/master/demos/forward.py, which will be very helpful in forwarding ports. You are encouraged to use paramiko.

## Example Output

```
$ python implant.py
[*] Dumping /etc/shadow
[!] Initializing john to crack 1 passwords
[*] New password: root/password
[*] Initializing SSH network worm with starting credentials
root/password
```

```
[*] Read targets file
[*] Attempting login to 10.1.0.2 with root/password
[*] Login to 10.1.0.2 succeeded
[*] No flag file found on 10.1.0.2
[*] Found servers file on 10.1.0.2
[+] Adding 10.2.0.1:22, 10.2.0.2:22, 10.2.0.6:1337 to global
target queue
[*] Searching for credentials on 10.1.0.2
[*] No SSH keys found
[*] Dumping /etc/shadow
[!] Initializing john to crack 4 passwords
[*] Successfully cracked 4/4 passwords
[+] New credentials: bob/password
[+] New credentials: alice/123456
[+] New credentials: srujan/iloveyou
[+] New credentials: manny/hello
[*] Forwarder setup on 10.1.0.2 to 10.2.0.2
[*] Attempting login to 10.2.0.2 with root/password
[*] Attempting login to 10.2.0.2 with bob/password
[*] Attempting login to 10.2.0.2 with alice/123456
[*] Attempting login to 10.2.0.2 with srujan/iloveyou
[*] Login to 10.2.0.2 succeeded
[*] Found servers file on 10.2.0.2
[*] No flag file found on 10.2.0.2
[*] No SSH keys found
[*] User srujan on 10.2.0.2 cannot access /etc/shadow
...
[*] Attempting login to 10.16.2.1 with chiapet/chiapet1
[*] Login to 10.16.2.1 succeeded, found servers file
[*] Found servers file on 10.16.2.1
[!] Flag file found on 10.16.2.1, contents: “Successful network
compromise!”
...
======================
Collected 6 /etc/shadow files
Cracked 15 passwords
Collected 4 SSH keys
Found 1 flag file
Accessed following IPs:
```

```
IP         Username   Password
10.1.0.2   root       password
10.2.0.1   srujan     iloveyou
10.2.0.4   unknown    unknown
10.2.0.2   root       password
...
```

Note that collecting SSH keys are extra credit.

To signify the end of execution, a computer will have only one file, the **flag** file. If you find the flag file, that computer has no more neighbors and you have finished that part of execution. To signify you have found the **flag** file, please print out the contents of the file.

## Notes

You can assume the following:

- There are no infinite loops. For example, a computer will not have an entry to itself.
- All machines are valid and have at least one login that is findable on the network.
- The order in which you print flags is not specified.
- Logins may be used more than once.
- **rockyou.txt** will be provided in the current directory of your program.
- Your implant will be running on a default Kali installation. The other machines on the network will be unspecified.
- Only the root user will be able to access /etc/shadow.
- You will need to log in with unprivileged users onto the machines. This is because those machines could be a 'link' between multiple networks, and you may not know any root credentials on that machine. Once you have acquired root access on that machine, you no longer need to continue trying accounts on it as it has already been fully compromised.
    - Remember, all accounts can see the **servers.txt** file in their home directory, but only root can get access to the /etc/shadow file.
- You should assume that no user has sudo access on the machine.