Exercise 06

April 09, 2023

Overview

This exercise provides hands-on experience with Firewalls to prevent Denial of Service attacks.

Learning Objectives

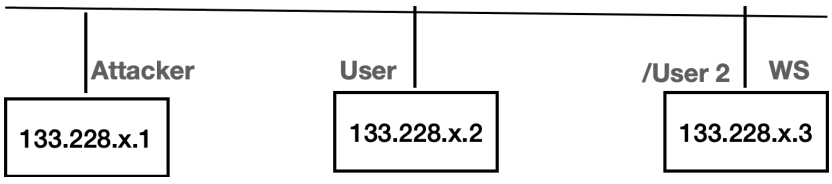
- Understand basics of DDoS Attack
- Understand basics of packet Filtering and Stateful Inspection firewall..
- Ability to implement the Firewall using ufw (Uncomplicated Firewall) in Ubuntu

Reading Material

1. Chapter 09, Principles of Computer Security, 4th ed, Stallings and Brown, Pearson.
2. Resources for implementing ufw (Uncomplicated Firewall)
    a. https://manpages.ubuntu.com/manpages/jammy/en/man8/ufw.8.html
    b. https://wiki.ubuntu.com/UncomplicatedFirewall
    c. https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands
3. Tools for launching DoS attacks
    a. https://www.kali.org/tools/hping3/
    b. https://nmap.org/book/man.html

Prerequisites and environment familiarity

- Familiarity with Linux and command line terminal usage.
- Familiarity with use `hping3` and its usage with different options
- Familiarity with `sysctl` and basic kernel network parameters configurations.
- You have access to 3 systems connected via a network
    - Kali Linux (attacker)
    - Ubuntu22 (entry point using GlobalProtect)
    - Ubuntu20 (victim or target machine)



Description

This exercise involves Launching various DoS attacks and then implement mechanisms to prevent these attacks using Uncomplicated firewall (`ufw`) and/or any other means.

Using `ufw`, the command line invocation does not support ICMP protocol. Any such rules related to ICMP should be specified in `/etc/ufw/before.rules` or `/etc/ufw/after.rules` and then reload the `ufw` configuration.

Assignment work details

Each attack should last at least 1 minutes. More the better

a. ICMP flood attack, and its prevention using `ufw`.
   The flood attack should be launched from specific source IP (attacker machine) and thus attack blocking using `ufw` should be done for that attacker IP rather than blocking all ICMP traffic. Ping should continue to work from other IP addresses but should be blocked only from attacker machine.

b. ICMP smurf attack and its prevention.
   Here the `ufw` command(s) should permit ICMP traffic from a specific IP address or subnet e.g. your own local network such as 133.228.x.0/24 but it should be blocked by default otherwise i.e. should be blocked from smurfed IP Addresses 133.228.y.0/24, where x<>y.

c. UDP flooding attack and its prevention.
   The `ufw` command (s) should prevent UDP flooding attacks from specific sources (attacker machine) and on to a specific destination port. **The destination port should be equal to last 4 digits of your UMBC Id**. All other UDP traffic should be permitted. For the same destination port, udp traffic should be permitted from other machines (e.g. Ubuntu22) than the attacker machine.

   Differentiate between **Deny** and **Reject**. In the case of Deny, there will be no response generated, whereas in the case of Reject, and ICMP error "*Port unreachable*" should be generated and sent to attacker.

d. HTTP Slowloris attack and its prevention.
   Issue a `ufw` limit command to allow limited http traffic from specific source (attacker). `Ufw` limit permits only 6 connections every 30 seconds (and this rate can't be changed at `ufw` level). Thus, generating any traffic at a higher rate should result in connection refused. For example, the fist command below for load generation should results in success where second command results in connection refused after few HTTP requests.

```
while true; do wget  http://<WS IP>; sleep 6; done
while true; do wget  http://<IS IP>; sleep 4; done
```

e. TCP SYN flooding attack and its prevention by implementing TCP SYN Cookie in Ubuntu.
   By default, Ubuntu systems are configured to handle SYN flooding attack by implementing SYN cookies. If the value of sysctl configuration parameter `net.ipv4.tcp_syncookies` is 1, it implies system is configured to implement SYN cookies. It initially responds in a normal way for SYN requests till its queue for SYN

responses as per the value of (`net.ipv4.tcp_max_syn_backlog`, which is `256` by default) becomes full and after that for any new connection request, it responds to TCP SYN request using SYN cookies. So, to verify that SYN cookies is working, you need to do the following.

i. Send 500 or more SYN flooding requests using `hping3` (from attacker) to Apache web server (victim machine) and after receiving SYN-ACK, do not send ACK message (3rd message in TCP handshake).

1. Configure `iptables` on attacker (Kali Linux) that drops the received SYN-ACK message e.g. using the command as below. Since this SYN-ACK message is not received, it won't generate TCP Reset or (Ack msg, 3rd message in TCP handshake). Thus TCP connection state on Victim should show SYN-RECV for such connections till its timeout.
   ```
   iptables -A INPUT -i eth1 -s <IP of WS> -p tcp --tcp-flags SYN,ACK,FIN,RST SYN,ACK -j DROP
   ```

2. Count the number TCP connections with state `SYN_RECV` on the victim machine (e.g. Web Server). You can use `netstat` command on victim system to check the number of connections e.g.
   ```
   netstat -nat | grep :80| grep SYN_RECV
   ```
   There should be <256 lines in the output. You can count the same using `wc`.

3. Remove the `iptables` rule as configured in step 1. To remove the rule, invoke the rule with `-D` option instead of `-A` option.

4. On the web server, create/upload a large file (size >1MB) e.g. any of your photo with size of 1MB or more in the directory `/var/www/html`. Let us say this file is `photo.jpg`

5. Make N (=500) or more concurrent HTTP Requests to web server to fetch this image file i.e. invoke the URL `http://<IP` of WS>`/photo.jpg`. Since both the attacker system and victim system are on same high bandwidth network, download will happen immediately. Use some mechanism that ensures that this URL fetching take time. For example, following invocation will fetch this file of 1MB size in 500 seconds.
   ```
   wget --limit-rate=2000 http://<IP of WS>/photo.jpg
   ```
   Use simple scripting program to make N(=500) concurrent invocations.

6. Count the number of TCP ESTABLISHED connection on web server. These should be N lines in the output of netstat. This indicates that TCP Syn cookies is working.
   ```
   netstat -nat | grep :80| grep ESTABLISHED
   ```

Explanation and Hints

a. Launch one attack at a time, verify its occurrence and then stop the attack
b. Configure attack prevention rules using `ufw`, launch the attacks and verify attack prevention.
c. Logs of `ufw` are generally written into /var/log/ufw.log

Assessment and Rubric

Please do submit the following

1. Readme.txt file which will contain the following information
    a. Name, university id and IP addresses of systems being used.
    b. Challenges faced and how did you address these.
    c. Summary of your overall learning.
    d. References. Any website/resource that you used to took help.

2. A file Attack.txt that describes mechanisms used to launch DoS attacks and corresponding configuration rules using firewalls to prevent such attacks. Any such attack prevention should be logged in ufw log file. Upload a file that describes ufw rules and their impact in preventing the attack via the log file.

3. A file showing the proof that attack was prevented. It could be tcpdump file or ufw log or script command output capture file showing the activities done to prevent the attack.

Rubric for assessment (20 marks)

a. Readme.txt file containing all the required information. If a proper readme file is not uploaded, then evaluation will result in 0 marks.
b. 2 marks for each attack launch and its prevention.

Note

Any plagiarism activity will result in penalties of being awarded 0 marks. If you are using the sample program as shown in the class, please attribute the same.

<end of Exercise 06>