Lab 1.2 - Single Packet Authorization Execution

Table of contents

1 Introduction

This lab is the second part of a brief tutorial for implementing Single Packet Authorization (SPA) with fwknop. Before completing anything in this lab, be sure to complete Lab 1.1 - Single Packet Authorization Setup.

2 Performing SPA for SSH Access

This portion will require you to go back and forth from client to server. As an overview, you will first test that the client can access the server via SSH. Then you will modify the firewall rules on the server to prevent all SSH connections and test it. Lastly, you will use fwknop on the client to send the SPA packets so that the firewall opens for the client to be able to ssh into it, but not anyone else.

1. On the client, test that you can ssh into the server. Be sure to use the appropriate username for the server.

   ```
   ssh -p $SSH_PORT username@$SERVER_IP
   ```

   This should successfully grant you access.

2. On the client, disconnect from the server with `ctrl + d`.
3. On the server, set a variable to represent the server's ssh port.

```
SSH_PORT=22  # reminder: no spaces around = sign
```

4. On the server, get the interface for the internet connection using `ip address`, `ip addr`, or `ifconfig`.

```
# run one of these. The output is similar for each
ip addr
ip address
ifconfig
```

   Recall that the interface is typically in the form `eth0` or `enp0s3`. Values may differ. Note what it is.

5. On the server, set another variable that holds the value for the interface.

```
INTERFACE=enp0s3
```

6. On the server, view the current firewall rules to see the default with the `iptables` command. The flag -L is used to list the rules, the flag -n is used to output the numeric values for items that could resolve to non-numeric values (ex: show port 22 as `22` instead of as `ssh`), and the -v flag is to get extra details (verbose).

```
sudo iptables -L -nv
```

   You should see information about the INPUT chain, the FORWARD chain, OUTPUT chain, and the FWKNOP_INPUT chain. INPUT, FORWARD and OUTPUT most likely have a policy of ACCEPT at this point.

   Additionally, the INPUT chain should have 1 rule with FWKNOP_INPUT as the target.

7. On the server, add the following firewall rule with `iptables`. This rule is appending to the INPUT chain so that all ssh traffic is dropped.

```
sudo iptables -A INPUT -i $INTERFACE -p tcp --dport $SSH_PORT -j DROP
```

8. On the server, view the firewall rules again.

```
sudo iptables -L -nv
```

   You should see that the INPUT chain has been modified with respect to port 22 and the interface you specified.

9. On the client, test ssh again. It should not allow you in. You can use `ctrl+c` to cancel the request when you think it has been too long. Be sure to use the appropriate username for the server.

```
ssh -p $SSH_PORT username@$SERVER_IP
```

10. On the client, use nmap to check if SSH is visible on the server.

```
sudo nmap -sS -p $SSH_PORT $SERVER_IP
```

It should show that the STATE is `filtered`.

11. On the client, Use fwknop to send an SPA packet to the server.

```
fwknop -n $SERVER_IP
```

12. On the server, check that the SPA packet was accepted by checking the status of the fwknop server.

```
sudo service fwknop-server status
```

You should see information concerning the acceptance of the SPA packet and a time limit for when it expires. For example, it may show the following:

```
Added access rule to FWKNOP_INPUT for x.x.x.x -> 0.0.0.0/0 tcp/22,
expires at 1746380800.
```

13. On the client, test ssh again to get access. Note that if the time limit expired, you will need to resend the SPA packet. Be sure to use the appropriate username for the server.

```
ssh -p $SSH_PORT username@$SERVER_IP
```

You should have access, however when the timelimit expires, you'll need to re-establish the connection.

14. On the client, if not already disconnected, disconnect from the server.

15. On the server, to prevent the client from being locked out when the time limit expires, lets modify the firewall rules.

    i. First, delete the rule we already have.

```
sudo iptables -D INPUT -i $INTERFACE -p tcp --dport $SSH_PORT -j DROP
```

You can confirm that it was deleted with `sudo iptables -L -nv`.

    ii. Next, add the following rule to allow related and established connections.

```
sudo iptables -A INPUT -i $INTERFACE -p tcp --dport $SSH_PORT \
-m state --state RELATED,ESTABLISHED -j ACCEPT
```

    iii. Re-add the rule to drop all ssh connections.

```
sudo iptables -A INPUT -i $INTERFACE -p tcp --dport $SSH_PORT -j DROP
```

3 Clean Up

Once you've completed the lab, you may want to consider reseting your firewall rules back to the default.

You can remove all the rules in all chains using the -F flag for flush.

```
sudo iptables -F
```