Department of Informatics

IPTABLES

Network Security

Diego Sempreboni

INTRODUCTION

In this lab we will use iptables as a host-based firewall to protect the local machine from network scanning. For this lab, you will have to work, at least, in pairs. The Virtual Machine (VM) of one team member will take the role of an **attacker**, and the VM of the other as **defender** (**victim** of the attack).

The goal of the lab is to protect the defender's machine against the following two scenarios:

- An attacker doing a **port scanning** against the defender.
- An attacker doing a **DoS attack** against the defender.

  For the first scenario, we will use NMap as described next. For the second scenario, we will use the DoS attack shown in one of the labs earlier. In this lab, we assume that the attacker has swept the network and knows the IP address of the victim. This can also be done with NMAP, although we will refrain from doing so to avoid congestion on the network. To directly find out the IP address of the victim's machine, type the following command in the defender's machine:

  (Ubuntu)

  ```
  # ifconfig
  ```

BASIC NETWORK SCANNING

There are different ways of doing a network scanning. The most basic one is using ICMP echo request that goes on top of UDP, although there are other types of scans over TCP. In this part of the lab you will be using basic UDP and TCP options to run a network scan. For these, we will mainly rely on ping and NMap. NMap is a very versatile tool that allow an attacker to map a network. Advanced students can optionally look at other more advanced options from NMap.

Nmap is one of the most popular tools **used for the enumeration of a targeted host**. Nmap can use scans that provide the **OS**, **version**, **and service detection** for individual or multiple devices. **Detection scans are critical to the enumeration process** when conducting penetration testing of a network. It is important to know where vulnerable machines are located on the network so they can be fixed or replaced before they are attacked

1. **Ping Sweeping**: Determine the host is up using ICMP echo and a TCP ACK.

```
# nmap -sP <victim_ip>
```

Try as well without NMap:

```
# ping <victim_ip>
```

2. **Port Scanning**: Determine if any (potentially vulnerable) services are available. There are different ways in which we can understand which types of ports there are open, such as TCP connect, TCP SYN, Stealth FIN, Xmas Tree, and Null, as well as UDP scans.

   - TCP connect `# nmap -sT <victim_ip>`
   - UDP Scanning `# nmap -sU <victim_ip>`

Task: Use some time to understand the difference between each type of scan.

SYSTEM FIREWALLING

We will next use IPTables to defend the machine against the aforementioned attacks.

1. **Understanding IPTables**

IPTables is a popular tool used to filter network packets in Linux systems. There are two key concepts to understand how IPTables work: **tables** and **chains**. Tables are a general abstraction of the different chains and there are currently five tables: **raw**, **filter**, **nat**, **mangle**, and **security**. For the purpose of this lab, we will mainly rely on the filter table. This table is the default one, where the actions of the firewall are defined.

Each table contains a number of chains (both built-in or user-defined). Chains contain a list of defined rules to match against incoming packets. The rules specify the actions to perform once a packet matches. This is called a **target**.

IPTables is used as follows:

> **iptables** [**-t** *table*] {**-A**|**-C**|**-D**} *chain rule-specification* **iptables** [**-t** *table*] **-I** *chain* [*rulenum*] *rule-specification* **iptables** [**-t** *table*] **-R** *chain rulenum rule-specification* **iptables** [**-t** *table*] **-D** *chain rulenum*
>
> **iptables** [**-t** *table*] **-S** [*chain* [*rulenum*]]

iptables [**-t** *table*] {**-F**|**-L**|**-Z**} [*chain* [*rulenum*]] [*options...*] **iptables** [**-t** *table*] **-N** *chain*
iptables [**-t** *table*] **-X** [*chain*]
iptables [**-t** *table*] **-P** *chain target*

iptables [**-t** *table*] **-E** *old-chain-name new-chain-name* rule-specification =
[*matches...*] [*target*]
match = **-m** *matchname* [*per-match-options*]
target = **-j** *targetname* [*per-target-options*]

We refer to the IPTables manual for further details
(http://ipset.netfilter.org/iptables.man.html).

2.- Delete any previous settings:

```
# iptables -F # iptables -F # iptables -F # iptables -F

INPUT
OUTPUT
FORWARD
```

3.- Blocking an incoming connection:

To block, for instance, all TCP packets from the attacker one can execute:

```
# iptables -I INPUT -p tcp -s <attacker_ip> -j DROP
```

where

- -I defines the chain for the default table
- -p specifies the protocol
- -s the source of the communication
- -j the target

Task: Probe the victim's machine from the attacker's machine as we did earlier. What has changed? What if the attacker moves to another machine?

4.- Default policy

Instead of blocking on a *case-by-case* basis, one can stablish a default policy and authorize expected network packets.

```
# iptables -P <chain> <target>
```

Task: Set a restrictive default policy and only allow the machine to:

i) Respond to **ping** only.
ii) allow outbound HTTP and HTTPS connections.
iv) allow inbound SSH from remote address.
v) allow security updates: ubuntu servers and DNS resolution. vi) allow already stablished connections
vii) log SSH attempts.

Hints:

- Understand what is the transport layer used by each protocol (i.e., UDP or TCP).
- Look at the list of well-known ports for each protocol.
- Ping packets are ICMP Type 8. •
- Understand the target you need (e.g., ACCEPT, DROP, or LOG).

5.- Persistency:

There is no option in iptables that would allow you to make your rules persistent. However, you can use the following commands to store the progress:

```
# iptables-save

# iptables-restore
```

You can store the progress into a file and restore later with the following:

```
# iptables-save > /etc/iptables.conf
# iptables-restore < /etc/iptables.conf
```

This can be automated by adding the iptables-restore to the *init* daemon. This is out of the scope of this lab.

BLOCKING DOS

Revisit the lab on *Denial of Service and IP Spoofing* you did a few weeks back and create an IPTables rule that would block the DoS attack.

Hint: look at the --hitcount option.