Outline:

I. Environment Setup

II. Today’s Task

III. Conclusion
Lab Objectives:

After this lab we expect all of you to know:

1. What is a FW with stateless rules and how it works
2. Set policies using iptables
3. Test the efficacy
4. Recommended Security Practises
I. Environment Setup:

➔ WinHost (Windows server 2008)
  • Putty

➔ UbuntuHost
  • Hping3 for port scanning

➔ Firewall (Debian)
  • Apache2 web server
  • Iptables
UbuntuHost ➔ UbuntuHost

Password : ubuntuhost

Check Settings :

- ifconfig

➔ Should be 192.168.1.6

Note : Access as root

➢ Type on terminal : sudo su
➢ Password : 123
WinHost

➡ Windows server 2008

Password : password@1

Check Settings :

- ipconfig

➡ Should be 192.168.1.1
Firewall

➔ Debian

Password : secclass

Check Settings :

- ifconfig

➔ Should be 192.168.1.2

Note : Login as root

➢ Type on terminal : su -
➢ Password : password@1
II. Today’s Task

1. **USING STATELESS RULES TO FILTER TRAFFIC**
   - Default Accept Policy on chains for the filter table
   - Block all ICMP echo(8) packets coming to the firewall
   - Default Drop Policy on chains for the filter table
   - Whitelist traffic for a specific Mac address
   - Allow access to tcp port 22 (ssh)

2. **ALLOWING SPECIFIC TCP FLAGS(SYN,FIN ACCEPT)**
   - Commands for accepting packets containing SYN & FIN

3. **NAT & PORT FORWARDING**
   - Redirect Traffic from port 8080 to common http port 80 on DMZ interface.
Firewall Basics

A Firewall is a perimeter network component that filters incoming or outgoing traffic to and from the network.
OSI vs TCP/IP Model

<table>
<thead>
<tr>
<th>OSI Model</th>
<th>TCP/IP Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>APPLICATION 7</td>
<td>FTP services 7</td>
</tr>
<tr>
<td>PRESENTATION 6</td>
<td>ANSI 6</td>
</tr>
<tr>
<td>SESSION 5</td>
<td>Controls conversations/Sessions (Dialog, Control) Integrity and Reliability Descriptive naming 5</td>
</tr>
<tr>
<td>TRANSPORT 4</td>
<td>fragmentation/sequencing of data Reliable delivery Error recovery Flow control Multiplexing (PORTS) 4</td>
</tr>
<tr>
<td>NETWORK 3</td>
<td>End to end delivery Logical addressing fragmentation/sequencing for MTU Routing 3</td>
</tr>
<tr>
<td>DATA-LINK 2</td>
<td>Physical addressing Error detection (FCS/CRC) Acknowledgements Packet/Frame header and trailer bridging 2</td>
</tr>
<tr>
<td>PHYSICAL 1</td>
<td>Media interface Transmission method Signal strength Topology 1</td>
</tr>
</tbody>
</table>

ENCAPSULATION

- DATA
- SEGMENT
- PACKET or DATA-GRAM
- FRAME
- Bit or Data-Stream
Port Communication

Communication via TCP/IP operates by **IP-Addresses** and **Ports**.

- Certain applications are associated with specific port numbers ranging from 0 to 65535.
- The ports below **1024** are **standardized** (standard ports), which are allocated to dedicated services, i.e.
  - 25 smtp
  - 80 http
  - 443 https
  - 22 ssh
Policies for Packet Filtering

There are 2 different strategies:

- **Deny every packet** (Only well defined kind of packets are allowed)
- **Allow every packet** (Only well defined kind of untrusted packets are discarded).

➔ **Reject VS Drop**:

**Reject**: The Packets will be discarded and an ICMP-Error message will be delivered to the sender.

**Drop**: The Packets will be discarded. Better choice, because:

➔ Less traffic,
➔ Some packets could be part of an attack
➔ An error message could contain useful information for an attacker
Iptables (Packet filter in Linux):

Three Chains:

--> **INPUT**: Filters traffic destined to fw machine itself
--> **OUTPUT**: Filters traffic generated by fw machine.
--> **FORWARD**: Filters traffic routed through the fw.

**NOTE**: “Accept is the default policy of iptables.”
Some handy rules:

➔ **Flush Tables:**
  ```
  iptables -F
  iptables -t nat -F
  ```

➔ **Drop Input, Output and Forward:**
  ```
  iptables -P INPUT DROP
  iptables -P OUTPUT DROP
  iptables -P FORWARD DROP
  ```

➔ **Check statistics & Rules:**
  ```
  iptables -t nat -L
  iptables -L -n -v
  ```

➔ **Forward ICMP packets from eth0 to eth1**
  ```
  iptables -A FORWARD -p ICMP -i eth0 -o eth1 -j ACCEPT
  ```

where

➔ **-p** = Protocol like **TCP, UDP and ICMP**,

➔ **-i** and **-o** flags respectively **input** and **output interfaces**.

➔ **-s** and **-d** are **source** and **destination**.
Let's Get Our Hands DIRTY!!

1. **USING STATELESS RULES TO FILTER TRAFFIC**
   - Default Accept Policy on chains for the filter table
   - Block all ICMP echo(8) packets coming to the firewall
   - Default Drop Policy on chains for the filter table
   - Whitelist traffic for a specific Mac address
   - Allow access to tcp port 22 (ssh)
1. **USING STATELESS RULES TO FILTER TRAFFIC**

→ **Default-Allow**

- *On Firewall Vm*, check that the rules are on “default Allow” (accepting all traffic) using `iptables -L -n`

```
root@StatelessFw:/home/secclass# iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination

Chain FORWARD (policy ACCEPT)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```

- Open terminal and ping from **WinHost & UbuntuHost** to the **Firewall**

```
Pinging 192.168.1.2 with 32 bytes of data:
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64
Reply from 192.168.1.2: bytes=32 time<1ms TTL=64

Ping statistics for 192.168.1.2:
   Packets: Sent = 3, Received = 3, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
```


- **Block all ICMP echo(8) packets coming to the server**
- Perform a continuous ping from **WinHost** terminal: `ping 192.168.1.2 -t`
- On **UbuntuHost** Implement the following rules:

```bash
root@StatelessFw:/etc# iptables -A INPUT -p icmp -d 192.168.1.2 --icmp-type 8 -j DROP
```

**Note**: ICMP (Internet Control Message Protocol) is an *error-reporting protocol*, it is *not* a transport protocol that sends data between systems. Any IP network device has the capability to send, receive or process ICMP messages.
Testing:

As it can be observed, initially there was a continuous flow of packets, but after the rules are implemented the ICMP packets are dropped.
Default Drop Policy on chains for the filter table

- From the terminal on the **Firewall**
- Type the following commands to set all policies to DROP from ACCEPT

```bash
root@StatelessFw:/home/secclass# iptables -P INPUT DROP
root@StatelessFw:/home/secclass# iptables -P OUTPUT DROP
root@StatelessFw:/home/secclass# iptables -P FORWARD DROP
```

- List the new policies using `iptables -L -n -v`

```bash
root@StatelessFw:~# iptables -L -n -v
Chain INPUT (policy DROP 0 packets, 0 bytes)
   pkts bytes target prot opt in  out  source     destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
   pkts bytes target prot opt in  out  source     destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
   pkts bytes target prot opt in  out  source     destination
```
- Open terminal on the **UbuntuHost** and ping the **Firewall**

```
root@hacking-VirtualBox:/home/hacking# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
11 packets transmitted, 0 received, 100% packet loss
time 10081ms
```

All Packets are lost

- Check traffic on the **Firewall** using `iptables -L -n -v`

```
root@StatelessFw:~ # iptables -L -n -v
Chain INPUT (policy DROP 96 packets, 8488 bytes)
    pkts bytes target     prot opt in     out     source                     destination

Chain FORWARD (policy DROP 0 packets, 0 bytes)
    pkts bytes target     prot opt in     out     source                     destination

Chain OUTPUT (policy DROP 0 packets, 0 bytes)
    pkts bytes target     prot opt in     out     source                     destination
```
Whitelist traffic from the WinHost's mac address

- **Check mac address of WinHost**: Type `ipconfig /all`

  - Host Name: WIN-RK6GKM526CU
  - Physical Address: 00-09-27-57-5F-20
  - IPv4 Address: 192.168.1.1 (Preferred)
  - NetBIOS over Tcpip: Enabled

- Define policy to allow outgoing traffic from the firewall:

  ```
  root@StatelessFw:/home/secclass# iptables -P OUTPUT ACCEPT
  ```

- Allow traffic for the Winhost’s mac address

  ```
  root@StatelessFw:/home/secclass# iptables -A INPUT -m mac --mac-source 08:00:27:57:5F:20 -d 192.168.1.2/32 -j ACCEPT
  ```
Testing:

- On **WinHost**, Open Putty located on the taskbar and connect to the **Firewall**:

  - **On WinHost**, Open Putty located on the taskbar and connect to the **Firewall**:

    - Login: `secclass`
    - Password: `secclass`

  - **Pings from UbuntuHost to Firewall vm doesn’t work**

```
root@hacking-VirtualBox:/home/hacking# ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2) 56(84) bytes of data.
^C
--- 192.168.1.2 ping statistics ---
12 packets transmitted, 0 received, 100% packet loss, time 11050ms
```
Allow access to tcp port 22 (ssh) from UbuntuHost

- Open Terminal on Firewall
- Flush the Iptables using `root@StatelessFw:/home/secclass# iptables -F`
- Allow access for 192.168.1.6

```
root@StatelessFw:/home/secclass# iptables -A INPUT -i eth0 -p tcp --dport 22 -s 192.168.1.6/32 -d 192.168.1.2/32 -j ACCEPT
root@StatelessFw:/home/secclass# iptables -L
Chain INPUT (policy DROP)
target prot opt source destination
ACCEPT tcp -- 192.168.1.6 StatelessFw tcp dpt:ssh

Chain FORWARD (policy DROP)
target prot opt source destination

Chain OUTPUT (policy ACCEPT)
target prot opt source destination
```
- Testing:

Test by Telnet 192.168.1.2 22 from UbuntuHost

```
root@hacking-VirtualBox:/home/hacking# telnet 192.168.1.2 22  
Trying 192.168.1.2...  
Connected to 192.168.1.2.  
Escape character is '^[}'.  
SSH-2.0-OpenSSH_6.7p1 Debian-5+deb8u2  
^C  
Connection closed by foreign host.
```
2. **FILTERING SPECIFIC TCP FLAGS( SYN, FIN ACCEPT) **

➔ **Accepting only packets containing SYN & FIN**

where

➔ **URG** - Urgent, **ACK** - Acknowledgement, **PSH** - Push, **RST** - Reset, **SYN** - Synchronize, and **FIN** - Finished are Flags contained in Transiting Packets

- on **Firewall**, insert the following rules:

```bash
root@StatelessFw:~# iptables -A INPUT -p tcp -m tcp --tcp-flags ALL SYN -j ACCEPT
root@StatelessFw:~# iptables -A INPUT -p tcp -m tcp --tcp-flags ALL FIN -j ACCEPT
```

Remember **Tcp** scheme
Testing:

- On **UbuntuHost**, Use **hping3** to view the dropped **tcp flags**:

  ```
  hping3 -c 1 -S 192.168.1.2
  ```

  ![hping3 output]

  - **I packet with SYN flag transmitted and received successfully**
  - **I packet with ACK flag dropped by firewall**
  - **I packet with PSH flag dropped by firewall**
  - **I packet with RST flag dropped by firewall**
  - **I packet with URG flag dropped by firewall**
  - **I packet with FIN flag transmitted and received successfully**
3. **NAT & PORT FORWARDING**

➔ **Redirect Traffic from port 8080 to common http port 80**

- Flush iptables with `iptables -F & iptables -t nat -F`

- Define all policies to **accept traffic**

```bashoot@StatelessFw:/home/secclass# iptables -P INPUT ACCEPT
root@StatelessFw:/home/secclass# iptables -P FORWARD ACCEPT
root@StatelessFw:/home/secclass# iptables -P OUTPUT ACCEPT
```

- Uncomment the following line in the `sysctl.conf` file

```bash
root@StatelessFw:~# gedit /etc/sysctl.conf
```

```bash
# Uncomment the next line to enable packet forwarding for IPv4
net.ipv4.ip_forward=1
```
- **check iptables** `iptables -L` & **check the nat policies by using** `iptables -t nat -L` 

```bash
root@StatelessFw:/home/secclass# iptables -t nat -L
Chain PREROUTING (policy ACCEPT)
target prot opt source                       destination
Chain INPUT (policy ACCEPT)
target prot opt source                       destination
Chain OUTPUT (policy ACCEPT)
target prot opt source                       destination
Chain POSTROUTING (policy ACCEPT)
target prot opt source                       destination
```

- **Insert the following rules to activate port redirection**:

```bash
root@StatelessFw:/home/secclass# echo 1 > /proc/sys/net/ipv4/ip_forward
root@StatelessFw:/home/secclass# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
root@StatelessFw:/home/secclass# iptables -A FORWARD -i eth0 -o eth1 -j ACCEPT
root@StatelessFw:/home/secclass# iptables -t nat -I PREROUTING --src 192.168.1.0/24 --dst 172.16.1.2 -p tcp --dport 80 -j REDIRECT --to-ports 8080
root@StatelessFw:/home/secclass# CONFIG IP NF NAT LOCAL=y
root@StatelessFw:/home/secclass# iptables -t nat -I OUTPUT --src 192.168.1.0/24 --dst 172.16.1.2 -p tcp --dport 80 -j REDIRECT --to-ports 8080
```
Testing:

- Open a browser on WinHost and Type “172.16.1.2:80”

- It Works!!
III. Conclusion

➔ **How can I protect my own PC**

- **Uninstall** all programs which are not permanently used.
- **Uninstall** all programs with well known security gaps e.g. Adobe Flash
- **Update your applications and operating systems as soon as stable updates are available**
- Invest in a good antivirus system *e.g. Kaspersky*
- Install a **personal firewall** (Freeware: ZoneAlarm)
- Encrypt your hard drive
- Scan all external usbs
- Use a trusted VPN service provider to encrypt your traffic
Best practices for firewall administrators

- Document all firewall rule changes.
- Install all access rules with minimal access rights. Eg. Avoid rules where the service field is 'ANY', it opens up 65,535 TCP ports as well as udp & icmp ports.
- Verify every firewall change against compliance policies and change requests.
- Remove unused rules from the firewall rule bases when services are decommissioned.
- Perform a complete firewall review at least twice per year.
Thanks!!