Hands-on Lab Description

ThoTh Lab

CS-CNS-20002 –
Network and Security: Testing Firewall and IDS

CONTENTS

Category:

```
CS-CNS: Computer Network Security
```

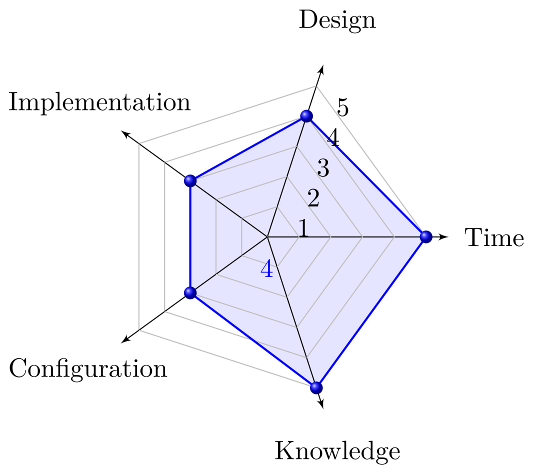
Objectives:

```
Understand vulnerability scanning and penetration testing tool NMap
Use NMap for firewall and IDS penetration testing
Use Metasploit to explore vulnerabilities that can pass through the firewall and IDS
Assess security strength and weakness of firewall and IDS
Provide pentesting report
```

Estimated Lab Duration:

```
Expert: 180 minutes
Novice: 600 minutes
```

Difficulty Diagram:



Difficulty Table.

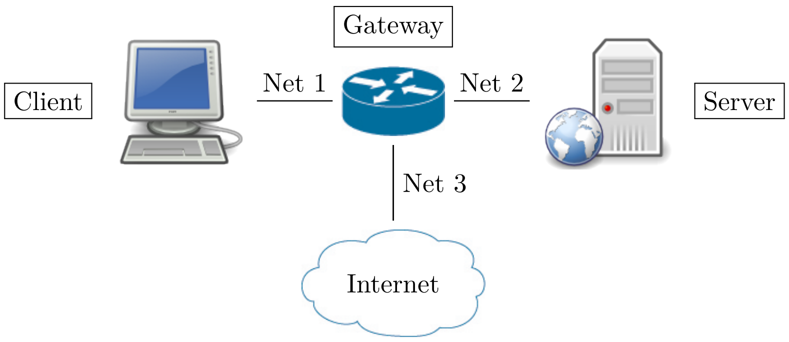
Measurements	Values (0-5)
Time	5
Design	4
Implementation	3
Configuration	3
Knowledge	5
Score (Average)	4

Required OS:

```
Linux: Ubuntu 18.04 LTS (Bionic Beaver)
Linux: Metasploitable 2
```

Lab Running Environment:

```
ThoTh Lab: https://thothlab.org
```



```
Client: Linux (Ubuntu 18.04 LTS - Pentesting Node)
Server: Linux (Metaspoitable 2 - Vulnerable server)
Gateway: Linux (Ubuntu 18.04 LTS - Running firewalls and IDS)
Network Setup:
Internet is connected through Net 3: 172.16.0.0/12
Client-side Net 1: 192.168.0.0/24
Server-side Net 2: 10.0.0.0/8
```

Lab Preparations:

```
Basic knowledge about computer networking (Reference Labs: CS-NET-00001 and CS-NET
    -00002)
Understand iptable filrewall (CS-CNS-00001) and snort intrusion detection system (CS-
    CNS-00003)
Know how to use Metasploit and understand Metasploitable 2 (CS-CNS-20010), and how to
     use NMap (CS-CNS-20001).
```

Lab Overview

In this lab, you will learn how to use penetration testing tools such as Nmap and Metasploit to perform firewall and IDS pentesting and write penetration testing report.

In summary, you will do:

- Use nmap to deploy various scanning approaches to test firewall configuration and setup.
- Design nmap scanning strategies to bypass IDS.
- Report security testing and evaluations.
- Practice attacks using Metasploit.

1 Introduction of Firewall and IDS Pentesting

Firewall, Intrusion Prevention Systems (IPS), and Intrusion Detection Systems (IDS) are only as robust as you configure them for your perimeter security needs. Where they reside on your network, how they interact, and improving Penetration (Pen) Testing services is key to having a strong defensive perimeter for your organization. IDS network security and IPS security, along with Firewall leveraging, can help provide this.

Cyberinfrastructures using prevention platforms require both regularly scheduled performance assessments and changing topology, etc. Most industry-standard institutes recommend even more frequent assessments. It is advised to change your current devices around at a certain frequent level. If they are always static, on-going cybercriminals learn how they always sit on your perimeter and eventually hack them after long periods in scanning for weaknesses.

When performing pentest against your own networks, basically, you are evading your own security systems. We usually refer to these self-targeted evading activities as ethical hacking and perform the pentesting as 'white-hats'. Pentesting helps understand the danger of real attackers. If you can sneak around a blocked portmapper port using Nmap direct RPC scanning, then so can the bad guys. It is easy to make a mistake in configuring complex firewalls and other devices. Many of them even come with glaring security holes that conscientious users must find and close. Regular network scanning can help find dangerous implicit rules before attackers do.

There are good reasons for evading IDSs as well. Product evaluation is one of the most common. If attackers can slide under the radar by simply adding an Nmap flag or two, the system does not offer much protection.

Pentesting tools such as Nmap and Metasploit offer many features for evading firewall rules or sneaking past IDSs. These capabilities allow 'bad guys' to misuse it for 'evil' purposes. However, from the administrator perspectives, mimicking attackers' behaviors and actions (maybe the best that they can do) to identify security holes before attackers do. This lab is by-no-mean the utterly and eagerly promoting Nmap and Metasploit. It simply introduces the beginning level of cyber defenders to start their network security endeavors in the area of penetration testing, since we know pentesting represents the art of cybersecurity with full of creativities and deep understanding of the given security system.

2 Pentest Goal, Setup, and Procedure

2.1 Goals

The goal of this pentest is to assess the firewall rule setup and IDS system. In this pentest, we apply the white-box testing strategy, in which a *white-box* tester has the access and knowledge levels of the system admin, including the knowledge of iptables firewall rules and snort rules setup. However, there are several constraints apply:

1. Goal: The goal of the pentest is to explore the configuration issues of iptables and snort, i.e., identify the system weakness due to configuration flaws. This means that the pen tester knows the iptables and snort configurations, and networking system setup. During the pen test, the pen tester can monitor the actions/reactions/output of iptables and snort. The pen tester tries to explore vulnerabilities focusing on two areas:

   (a) System configurations: firewall setup such as open/close/filter/unfilter ports, reachable services or nodes, firewall performance measurements, etc.; IDS setup such as detection threshold setup, success, and missed attack scenarios, etc.

   (b) Vulnerabilities: There is one Metasploitable-2 VM running behind the firewall. The pen tester is required to assess the security performance of the firewall and IDS on successfully stop and/or detect deployed attacks.

2. Location: The pen tester explores the configuration vulnerabilities from outside, i.e., on the client network.

3. Assessment: The pen tester needs to develop a pentest report based on the provided template, and provide suggested changes or remediation of firewall and IDS rules to maximally improve the system security. Note that the Metasploitable 2 is used for assessing the firewall and IDS. Note that your pentest focus is to inspect and assess the security strength of the firewall and IDS. Please do not try to resolve any vulnerable issues running on Metasploitable 2, which is purposely designed for pentesting.

4. Tools: The pen tester can use various tools provided in the given client VM such as Nmap, hping3, MSF (Metasploit framework), tcpdump, etc. It should be sufficient to use these tools for this lab. However, you can use or develop your own tools and testing scripts to perform the security testing.

2.2 System Setup

The pentesting system setup is presented in Figure CS-CNS-20002.1.

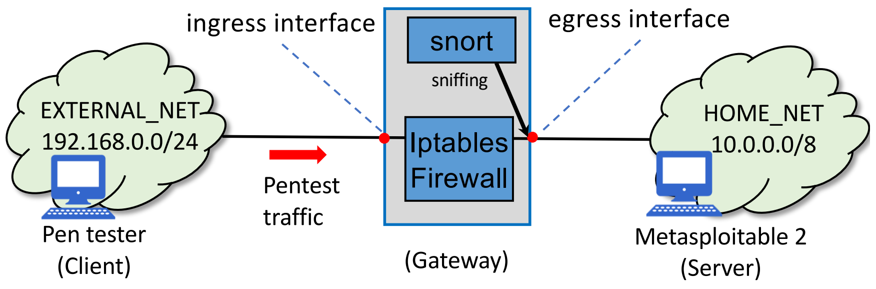


Figure CS-CNS-20002.1
Pentest System Setup.

In the following examples presented in this lab, we use the network setup in below, and you need to change the provided IP addresses accordingly in your lab running environment.

- Client: 192.168.0.7
- Gateway: 192.168.0.3, 10.0.0.4
- Server: 10.0.0.6

For each node setup and tools, we summarize as follows:

1. Client Node (Ubuntu 18.04): You should perform your pentesting tasks on the client machine. You can run various tools such as ping, traceroute, nmap, hpings, msfconsole, etc., on the client node.

2. Gateway Node (Ubuntu 18.04): You need to run iptables firewall and snort to protect the server node based on provided iptables and snort configuration files. You can to download the iptables configuration files and snort rule files using the following commands:

```
$    wget
       https://gitlab.thothlab.org/thoth-group/ThoThLabResource/raw/master/lab-cs-cns-20002.zip
$    unzip lab-cs-cns-20002.zip
$    cd lab-cs-cns-20002
```

   The firewall script template files "*rc.firewall.pentest*" and "*rc.firewall.reset*" are located in the folder 'lab-cs-cns-20002'. They are shell scripts. To make them executable, you can change their permission by (refer to more details in the lab CS-SYS-00001 on Linux file permission):

```
$    sudo chmod 755 rc.firewall.pentest % this will change to the file to green
       when you show 'ls -l' command
$    sudo chmod 755 rc.firewall.reset % this script change the iptables to default
       blacklist policies
```

   Note that you need to change the parameter setup at the beginning of the *rc.firewall.pentest* and *rc.firewall.reset* based on your lab running environment, i.e., three interfaces and IP addresses. Also, make sure you have set up your default GW properly on each VM. To run the script files, you can:

```
$    sudo ./rc.firewall.pentest  % run the firewall script to test firewall rules
$    sudo ./rc.firewall.reset  % run this script to clear firewall rules and reset
       it to apply the blacklist policy
```

   The snort rule file is *local.rules*. To run snort, you need to do the following actions:

   (a) Replace the */etc/snort/rules/local.rules* with the provided local.rules file.

```
$ cp local.rules /etc/snort/rules/local.rules
```

   (b) Comment out all the rule files in the */etc/snort/snort.conf*, except the *local.rules*. To this end, you can run the following command to comment out all the include statement, and then later uncomment the line that includes *local.rules*:

```
$ sudo sed -i 's/include \$RULE_PATH/#include \$RULE_PATH/'
    /etc/snort/snort.conf
```

   (c) Run the snort as follows:

```
$ sudo snort -c /etc/snort/snort.conf -A console -q -i ens5 % assuming ens5 is
    the interface on the server-side.
```

3. Server Node (Metastploitable 2): Refer to CNS-20010 (Network and Security Tool: Metasploit (CLI)) for more details on Metasploitable 2, especially the vulnerable services running on it. You do not need to make any changes or adjustments on the provided Metasploitable 2 virtual machine in this lab.

4. In this lab, iptables, snort, Nmap, and Metasploit are four major security services and tools that you will use. Please refer to the following labs on how to operate them in more details:

   - Iptalbes firewall (LAB-CS-CNS-00001)
   - Snort IDS (LAB-CS-CNS-00003)
   - Nmap (LAB-CS-CNS-20001)
   - Metasploit Framework (LAB-CS-CNS-20010)

2.3 Procedure

The pentest procedure of this lab is guided by the book:

```
Nmap network scanning authored by Gordon ‘‘Fyodor’’ Lyon, which is available at
    https://Nmap.org/book/.
```

In particular, you will follow the pentest approaches and analysis strategies given in the *Chapter 10 "Detecting and Subverting Firewalls and Intrusion Detection Systems"*, which is available at `https://Nmap.org/book/firewalls.html`. The required Tasks 1-4 are mapped to four sections of the chapter, respectively. For better guidance of the reading and task preparation, the *task-chapter* mapping is provided in Table CS-CNS-20002.1.

Table CS-CNS-20002.1
The mapping of Tasks and Chapter/Sections

Tasks	Section and URL
1	Determining Firewall Rules `https://Nmap.org/book/determining-firewall-rules.html`
2	Determining Firewall Rules `https://Nmap.org/book/firewall-subversion.html`
3	Determining Firewall Rules `https://Nmap.org/book/subvert-ids.html`
4	Detecting Packet Forgery by Firewall and Intrusion Detection Systems `https://Nmap.org/book/firewall-ids-packet-forgery.html`

In the following presented Task 1 to Task 4, they summarized the pentesting procedures and considerations described in chapter 10 of the Nmap network scanning book. For Tasks 1-4, there are suggested 14 analysis points. For each analysis point, you need to provide the following demonstrations/analysis:

1. Actions: Describe and demo your actions such as issues commands and interactive responses.
2. Outputs/results: Based on actions, what is the output or results generated from the actions.
3. Analysis: Your interpretations and analysis of the outputs and results, including facts and security analysis with your rationale.
4. Remediation Suggestions: your suggested changes of the firewall and IDS considering the pros and cons of security improvements and interruption of impacted services.

Task 5 is based on the Metasploit Framework lab (LAB-CS-CNS-20010), you need to demonstrate how to successfully deploy attacks through the firewall. In this task, you need to successfully exploit at least **two** vulnerabilities to compromise the targeted service/application, e.g., stop the service, gain access to the command shell, modify data, or service parameters, steal critical data, modify system configuration/setup. To this end, you need to present the following information for each successful attack:

1. How to identify the vulnerability?
2. The steps of using Metasploit to explore the vulnerability.
3. Suggest the countermeasure or mitigation solutions based on given firewall rules (*rc.firewall.pentest*) and IDS detection rules (*rc.rules*).

2.4 Suggestions on how to provide an instructive and comprehensive pentest?

- To provide a comprehensive and in-depth analysis, you are recommended to read Chapter 10 "Detecting and Subverting Firewalls and Intrusion Detection Systems" (`https://Nmap.org/book/firewalls.html`) carefully while doing each of your analysis point.

- *rc.firewall.pentest* is the firewall setup that you want to perform the pentest. To have better understanding of various scans, you can use *rc.firewall.reset* to clear the firewall rules and set it to blacklist (you need to change your scanning target directly to the Metasploitable-2 VM), and compare scanning responses with the *rc.firewall.pentest* firewall rules. In this way, you can observe results w/o firewall rules to interpret the scanning outcomes.
- Nmap and Metasploit Framework are installed on both the Client and Gateway VMs. Thus, you can also compare the scanning results on the Client vs on the Gateway, which can help you determine if the firewall successfully defend a scan/attack.
- Using tcpdump to monitor the traffic from both ingress and egress ports of the gateway is always the best approach to help you understand the actions performed on the Gateway.
- Finally, you are not restricted by the enumerated scanning approaches and provided hints. Please jump out or the box and try to provide surprising results with your own approach and in-depth analysis, since pentest is the art of cyberdefense, and you need to be creative.

3 Task 1 Determine Firewall Rules

3.1 Determining Stateful or Stateless Firewall

One helpful feature of the TCP protocol is that systems are required by RFC 793 to send a negative response to unexpected connection requests in the form of a TCP RST (reset) packet. The RST packet makes closed ports easy for Nmap to recognize. Filtering devices such as firewalls, on the other hand, tend to drop packets destined for disallowed ports. In some cases they send ICMP error messages (usually port unreachable) instead. Because dropped packets and ICMP errors are easily distinguishable from RST packets, Nmap can reliably detect filtered TCP ports from open or closed ones, and it does so automatically.

The ACK scan sends TCP packets with only the ACK bit set. Whether ports are open or closed, the target is required by RFC 793 to respond with a RST packet. Firewalls can block the probe, i.e.,

```
iptables ... -j DROP
```

and the firewall usually makes no response or send back an ICMP destination unreachable error. This distinction allows Nmap to report whether the ACK packets are being filtered. The set of filtered ports reported by an Nmap ACK scan is often smaller than for an SYN scan against the same machine because ACK scans are more difficult to filter. Stateless firewall can block incoming SYN packets (without the ACK bit set) is an easy way to do this, but it still allows any ACK packets through. Blocking those ACK packets is more difficult, because they do not tell which side started the connection.

Run SYN scan and ACK scan and compare difference.

```
$ Nmap -sS -T4 192.168.0.3     % SYN Scan
$ Nmap -sA -T4 192.168.0.3     % ACK Scan
```

To block unsolicited ACK packets (as sent by the Nmap ACK scan), while allowing ACK packets belonging to legitimate connections, firewalls must statefully watch every established connection to determine whether a given ACK is appropriate.

Please check the provided iptables firewall rules and check where the stateful firewall policies kicked in. These stateful firewalls are usually more secure because they can be more restrictive. Blocking ACK scans is one extra available restriction. The downsides are that they require more resources to function, and a stateful firewall reboot can cause a device to lose state and terminate all established connections passing through it.

Analysis Point 1: the key analysis point of this task is to check the feedback of Nmap scans with different scanning options to understand the implication of Nmap outputs: filtered, unfiltered, closed, open. What are the implications of the firewall setup according to these feedback.

3.2 Interpret Firewall Rules by Using various Scanning Approaches

3.2.1 Standard SYN Scan

Systems running TCP protocol is required by RFC 793 to send a negative response to unexpected connection requests in the form of a TCP RST (reset) packet.

```
$ Nmap -sT -T4 192.168.0.3     % SYN Scan
```

Analysis Point 2: the key analysis point of this task is how to infer the deny-by-default firewall policy based on the scanning responses from the firewall.

3.2.2 ACK Scan

The ACK scan sends TCP packets with only the ACK bit set. Whether ports are open or closed, the target is required by RFC 793 to respond with a RST packet.

```
$ Nmap -sT -T4 192.168.0.3     % SYN Scan
```

Analysis Point 3: the key analysis point of this task is how to infer stateful or stateless firewall policies setup based on the scanning responses from the firewall.

3.2.3 Compare the responding time: standard scan, SYN scan, and ACK scan

Run the following command to observer the system responding time.

```
$ Nmap -sT -T4 192.168.0.3     % Standard Scan
$ Nmap -sS -T4 192.168.0.3     % SYN Scan
$ Nmap -sA -T4 192.168.0.3     % ACK Scan
```

Analysis Point 4: What is your observation of the responding time of each type of scans, and what is the reason and implications?

3.2.4 IP ID (Idle) scanning

1. Find at least one accessible (open or closed) port of one machine on the internal network. The machine chosen should have little network traffic to avoid confusing results. (Most Linux system does not work).

2. Verify that the machine has predictable IP ID sequences. For example, targeting a Windows XP (e.g., 192.168.0.3), the Nping options request that five SYN packets be sent to port 80, one second apart.

   ```
   $ nping -c 5 --delay 1 -p 80 --tcp 192.168.0.3
   ```

   Typical predictable response:

   ```
   Starting Nping ( http://Nmap.org/nping )
   SENT (0.0210s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=48089iplen=40
       seq=136013019 win=1480
   RCVD (0.0210s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4900iplen=40
       seq=0 win=0
   SENT (1.0220s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=41250 iplen=40
       seq=136013019 win=1480
   RCVD (1.0220s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4901iplen=40
       seq=0 win=0
   SENT (2.0240s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=10588 iplen=40
       seq=136013019 win=1480
   RCVD (2.0250s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4902iplen=40
       seq=0 win=0
   ```

```
SENT (3.0270s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=55928 iplen=40
    seq=136013019 win=1480
RCVD (3.0280s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4903iplen=40
    seq=0 win=0
SENT (4.0300s) TCP 192.168.0.21:42091 > 192.168.0.40:80 S ttl=64 id=3309 iplen=40
    seq=136013019 win=1480
RCVD (4.0300s) TCP 192.168.0.40:80 > 192.168.0.21:42091 RA ttl=128 id=4904iplen=40
    seq=0 win=0

Max rtt: 0.329ms | Min rtt: 0.288ms | Avg rtt: 0.300ms
Raw packets sent: 5 (200B) | Rcvd: 5 (230B) | Lost: 0 (0.00%)
Tx time: 4.00962s | Tx bytes/s: 49.88 | Tx pkts/s: 1.25
Rx time: 5.01215s | Rx bytes/s: 45.89 | Rx pkts/s: 1.00
Nping done: 1 IP address pinged in 5.03 seconds
```

3. Start a flooding to a chosen host such as 192.168.0.7. In this example, you want to test if the incremental ID works properly. Do not use any of the address from the nping node.

   ```
   $ nping -S 192.168.0.7 --rate 10 -p 80 -c 10000 --tcp 192.168.0.3
   ```

   Then, run

   ```
   $ nping -c 5 --delay 1 -p 80 --tcp 192.168.0.3
   ```

   The example output is:

   ```
   Starting Nping ( http://Nmap.org/nping )
   SENT (0.0210s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=61263iplen=40
       seq=292367194 win=1480
   RCVD (0.0220s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5755iplen=40
       seq=0 win=0
   SENT (1.0220s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=30096iplen=40
       seq=292367194 win=1480
   RCVD (1.0220s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5766iplen=40
       seq=0 win=0
   SENT (2.0240s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=26815iplen=40
       seq=292367194 win=1480
   RCVD (2.0240s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5777iplen=40
       seq=0 win=0
   SENT (3.0260s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=49116iplen=40
       seq=292367194 win=1480
   RCVD (3.0270s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5788iplen=40
       seq=0 win=0
   SENT (4.0290s) TCP 192.168.0.21:1781 > 192.168.0.40:80 S ttl=64 id=2916iplen=40
       seq=292367194 win=1480
   RCVD (4.0300s) TCP 192.168.0.40:80 > 192.168.0.21:1781 RA ttl=128 id=5799iplen=40
       seq=0 win=0

   Max rtt: 0.342ms | Min rtt: 0.242ms | Avg rtt: 0.272ms
   Raw packets sent: 5 (200B) | Rcvd: 5 (230B) | Lost: 0 (0.00%)
   Tx time: 4.00853s | Tx bytes/s: 49.89 | Tx pkts/s: 1.25
   Rx time: 5.01106s | Rx bytes/s: 45.90 | Rx pkts/s: 1.00
   Nping done: 1 IP address pinged in 5.03 seconds
   ```

It shows that the IP IDs are increasing by roughly 11 per second instead of one. The target (192.168.0.3) is receiving our 10 forged packets per second and responding to each of them.

4. Repeat step 3 using spoofed addresses that you suspect may be allowed through the firewall. Try addresses behind their firewall and the RFC 1918 private networks such as 10.0.0.0/8, 192.168.0.0/16, and 172.16.0.0/12. This technique's result is a list of source address netblocks that are permitted through the firewall and those that are blocked.

Analysis Point 5: what you have done to determine if the system provides you needed setup to deploy the idle scan?

3.2.5 UDP Scanning

UDP is often more difficult because the protocol does not provide acknowledgment of open ports like TCP does. Many UDP applications will ignore unexpected packets, leaving the scanner unsure whether the port is open or filtered. Try the following command:

```
$ Nmap -sU -p50-59 192.168.0.3 % UDP scan
$ Nmap -sV -sU -p50-59 192.168.0.3 % UDP scan with version detection
```

Nmap shows "open/filtered" state if no response is received from a port.

Analysis Point 6: what is the implication of "open/filtered" states of the UDP scan?

4 Task 2: Bypassing Firewall Rules

4.1 Use TCP flags

Please refer to Lab CNS-20001 (Network and Security Tool: Nmap) and `https://Nmap.org/book/scan-methods.html` for more details on how to use Nmap to perform port scanning.

Try the following commands and observe their output differences:

```
$ Nmap -sT -p1-100 192.168.0.3 % Full scan
$ Nmap -sS -p1-100 192.168.0.3 % SYN scan
$ Nmap -sA -p1-100 192.168.0.3 % ACK scan
$ Nmap -sF -p1-100 192.168.0.3 % FIN scan
$ Nmap -sX -p1-100 192.168.0.3 % XMAS scan
$ Nmap -sN -p1-100 192.168.0.3 % Null scan
$ Nmap -sM -p1-100 192.168.0.3 % Maimcon scan
$ Nmap -sO -p1-100 192.168.0.3 % IP protocol scan
```

Analysis Point 7: Based on the scanning results, response to the following requests:

1. Provide a comparison table to show the key response features of these scans and their performance (i.e., response time). Address pros and cons of using these different types of scans concerning different firewall setup.
2. Based on the scanning results, can you further deploy idle scan and FTP Bounce scan? Show your analysis and exploration results.

4.2 Manipulate Source Port

Nmap uses the -g and –source-port options to manipulate source ports of transmitted packets. Compare the following two scanning results:

```
$ Nmap -sS -vv -Pn 192.168.0.3
$ Nmap -sS -vv -Pn -g 80 192.168.0.3
```

Analysis Point 8: What if you change the scanning target from 192.168.0.3 to 10.0.0.6, can you explain the differences?

4.3 Firewall Subversion

Try the following two pairs of commands:

```
$ Nmap -n -sn -PE -T4 10.0.0.0/24
$ Nmap -vv -n -sS -T4 -Pn --reason 10.0.0.0/24
$
$ Nmap -vv -n -sn -PE -T4 --packet-trace 10.10.10.6
$ Nmap -vv -n -sn -PE -T4 --packet-trace 192.168.0.3
```

Analysis Point 9: Based on the scanning results, can you describe your inference or findings?

5 Task 3: Subverting Intrusion Detection Systems

5.1 Identifying IDS from Name Convention

Run the following command to observe the identified host information.

```
$ Nmap -sS -sV -T4 -p1-100 192.168.0.3
```

Analysis Point 10: Describe if you can identify information for a running intrusion detection system?

5.2 Observe sudden firewall changes and suspicious packets

Many intrusion detection systems have morphed into what marketing departments label intrusion prevention systems. Some can only sniff the network like a normal IDS and send triggered packet responses. The best IPS systems are inline on the network to restrict packet flow when suspicious activity is detected. For example, an IPS may block any further traffic from an IP address that they believe has port scanned them or attempted a buffer overflow exploit. Attackers are likely to notice this if they port scan a system and cannot connect to the reported open ports. Attackers can confirm that they are blocked by trying to connect from another IP address.

Suspicious response packets can also tip-off that an IDS has flagged an attacker's actions. In particular, many IDSs that are not inline on the network will forge RST packets to tear down connections. Ways to determine that these packets are forged are covered in the section called "Detecting Packet Forgery by Firewall and Intrusion Detection Systems". For example, the Linux iptables system, which offers many methods for rejecting undesired packets by using:

```
--reject-with type
```

The type given can be icmp-net-unreachable, icmp-host-unreachable, icmp-port-unreachable, icmp-proto-unreachable, icmp-net-prohibited or icmp-host-prohibited, which return the appropriate ICMP error message (port-unreachable is the default). The option tcp-reset can be used on rules which only match the TCP protocol: this causes a TCP RST packet to be sent back. This is mainly useful for blocking ident (113/tcp) probes, which frequently occur when sending mail to broken mail hosts (which won't accept your mail otherwise).

Forging RST packets by firewalls and IDS/IPS is not particularly common outside of port 113. It can be confusing to legitimate network operators and allows scanners to immediately move on to the next port without waiting on the timeout caused by dropped packets. Nevertheless, it does happen. Such forgery can usually be detected by careful analysis of the RST packet compared to other packets sent by the machine. Refer to `https://Nmap.org/book/firewall-ids-packet-forgery.html` for more information.

Analysis Point 11: you can purposely alter some iptables rules and observe the difference and how the changes can impact your inferences about the firewall setup.

5.3 Unexplained TTL jumps

One way to detect certain IDSs is to watch for unexplained gaps (or suspicious machines) in traceroutes. While most operating systems include a traceroute command, Nmap offers a faster and more effective alternative with the –traceroute option.

```
$ Nmap --traceroute 10.0.0.6
```

Or you can use ping command:

```
$ ping -R 10.0.0.6
```

Analysis Point 12: This approach may only detect inline IDSs instead of those that passively sniff the network without being part of the route. Additionally, inspect iptables rules and output from Nmap and ping, and explain why the above-issued commands cannot return the path information.

5.4 Stealthy Scan

The following techniques are used to avoid being detected by the IDS (snort). Refer to `https://Nmap.org/book/performance.html` for more information on how to use Nmap performance options to execute stealthy Nmap scans.

Analysis Point 13: Your snort can detect many scanning activities, based on the given snort rules, how to configure your scanning parameters in the following given strategies: from *slow down* to *DNS proxying*, which ones you can deploy in our testing environment. You need to check the rules specified in the given snort *local.rules*. Based on the following given Nmap scanning methods, design new Nmap scanning commands to avoided being detected by snort rules:

```
$ Nmap -sT -p1-100 192.168.0.3 % Full scan
$ Nmap -sS -p1-100 192.168.0.3 % SYN scan
$ Nmap -sA -p1-100 192.168.0.3 % ACK scan
$ Nmap -sF -p1-100 192.168.0.3 % FIN scan
$ Nmap -sX -p1-100 192.168.0.3 % XMAS scan
$ Nmap -sN -p1-100 192.168.0.3 % Null scan
$ Nmap -sM -p1-100 192.168.0.3 % Maimcon scan
$ Nmap -sO -p1-100 192.168.0.3 % IP protocol scan
```

5.4.1 Slow down

Nmap offers several canned timing modes that can be selected with the -T option to accomplish this. For example, the -T paranoid option causes Nmap to send just one probe at a time, waiting five minutes between them. A large scan may take weeks, but at least it probably will not be detected. The -T sneaky option is similar, but it only waits 15 seconds between probes. Rather than specify canned timing modes such as sneaky, timing variables can be customized precisely with options such as –max-parallelism, –min-rtt-timeout, and –scan-delay.

Issue the following command and check if you can avoid being detected by the snort.

```
$ Nmap --scan-delay TIME-Value -p1-100 10.0.0.6
```

In the above example, the TIME-Value (e.g., 1075ms) is to be determined by you to avoid being detected by snort. For example, 1075ms is to ensure that Nmap waits 1.075 seconds between sending probes. The intuitive choice might be a one-second wait between packets to avoid 15 packets in 15 seconds, but that is not enough. Only 14 waits between sending the first packet and the fifteenth, so the wait must be at least 15/14, or 1.07143 seconds. You can inspect the snort threshold set up to find proper scan intervals.

5.4.2 Scatter probes across networks rather than scanning hosts consecutively

Nmap offers the –randomize-hosts option, which splits up the target networks into blocks of 16384 IPs, then randomizes the hosts in each block.

5.4.3 Fragment packets

IP fragments can be a major problem for intrusion detection systems, particularly because the handling of oddities such as overlapping fragments and fragmentation assembly timeouts are ambiguous and differ substantially between platforms. Thus, the IDS often has to guess at how the remote system will interpret a packet.

An Nmap scan will use tiny IP fragments if the -f is specified. By default, Nmap will include up to eight bytes of data in each fragment, so a typical 20 or 24 byte (depending on options) TCP packet is sent in three tiny fragments. Every instance of -f adds eight to the maximum fragment data size. So -f allows up to 16 data bytes within each fragment. Alternatively, you can specify the –mtu option and give the maximum data bytes as an argument. The –mtu argument must be a multiple of eight and cannot be combined with the -f option.

Fragmentation is only supported for Nmap's raw packet features, including TCP and UDP port scans (except connect scan and FTP bounce scan) and OS detection. Features such as version detection and the Nmap Scripting Engine generally don't support fragmentation because they rely on your host's TCP stack to communicate with target services.

5.4.4 Using Decoys

Decoys are added with the -D option. The argument is a list of hosts, separated by commas. The string ME can be used as one of the decoys to represent where the true source host should appear in the scan order. Otherwise, it will be a random position.

5.4.5 Port scan spoofing

Specify -S followed by a source IP, and Nmap will launch the requested port scan from that given source. No useful Nmap results will be available since the target will respond to the spoofed IP, and Nmap will not see those responses. IDS alarms at the target will blame the spoofed source for the scan. This can be useful for framing innocent parties, casting doubt in the administrator's mind about his IDS accuracy.

5.4.6 Idle scan

The idle scan is a clever technique that allows for spoofing the source IP address, as discussed in the previous section, while still obtaining accurate TCP port scan results. This is done by abusing the properties of the IP identification field as implemented by many systems. However, most Linux systems are immune to idle scan attacks.

5.4.7 DNS proxying

Nmap performs reverse-DNS resolution by default against every responsive host. These DNS lookup probes could be detected. Even something as unintrusive as a list scan (-sL) could be detected this way. The probes will come from the DNS server configured for the machine running Nmap. The most effective way to eliminate this risk is to specify -n to disable all reverse DNS resolution. You can specify one or more of those name servers to the –dns-servers option of Nmap, and all rDNS queries will be proxied through them.

6 Task 4 Detecting Packet Forgery by Firewall and Intrusion Detection Systems (Optional)

Read through the chapter "Detecting Packet Forgery by Firewall and Intrusion Detection Systems" at `https://Nmap.org/book/firewall-ids-packet-forgery.html`, and inspect each of the discussed topics and see if you can design any Nmap related tricks to detect forged packets from firewall and IDS.

Analysis Point 14: This task is optional. Bonus points will be considered for successfully demonstrated good tricks/solutions.

7 Task 5: Explore Vulnerabilities

The scanning results derived from task 1 to task 3 allows you to identify potential vulnerabilities to explore. In this task, you need to use Metasploit Framework (msf) to deploy at least two successful attacks, such as stop the service, gain access to the command shell, modify data, or service parameters, steal critical data, modify system configuration/setup. Note that scanning and identify vulnerabilities will not be counted. You can refer to the Lab CNS-20010 "Network and Security Tool: Metasploit (CLI)" to demonstrate your successful exploitation.

8 Deliverable

You should submit report via ThoTh Lab portal.

9 Lab Assessments (100 points)

Lab assessment for accomplishing Tasks 1-5 depends on the following facts:

1. (70 points) Task 1-3. You need to clearly present the following description for each task and each hint related pentest method.
   (a) Actions: Describe and demo your actions such as issues commands and interactive responses.
   (b) Outputs/results: Based on actions, what is the output or results generated from the actions.
   (c) Analysis: Your interpretations and analysis of the outputs and results, including facts and security analysis with your rationale.
   (d) Remediation Suggestions: your suggested changes of the firewall and IDS considering the pros and cons of security improvements and interruption of impacted services.
2. (Bonus points) Task 4.
3. (30 points) Task 5, each demonstrated successful attacks, will be counted for 15 points. You need to present the following information for each successful attack:
   (a) Present how to identify the vulnerability.
   (b) Present steps of using Metasploit to explore the vulnerability.
   (c) Explain the countermeasure or mitigation solutions based on firewall/IDS solutions.

10 Reference

```
Book "Nmap network scanning by Gordon "Fyodor" Lyon:
https://Nmap.org/book/

Detecting and Subverting Firewalls and Intrusion Detection Systems:
https://Nmap.org/book/firewalls. html

Top 1,000 TCP and UDP ports (Nmap default):
https://nullsec.us/top-1-000-tcp-and-udp-ports-Nmap-default/

Detecting Packet Forgery by Firewall and Intrusion Detection Systems
https://Nmap.org/book/firewall-ids-packet-forgery.html
```