

# **Lab 04: Network Reconnaissance & Firewall Evasion**

**Course:** Cyber Defense

**Report Prepared By:** [HASSANI, Mohammad Salem]

**Date:** [2025-12-15]

**Institution:** [Asen Grozdanov]

---

## **1. Lab Overview & Objectives**

In this lab, students explored the Reconnaissance and Evasion phases of the cyber kill chain from the perspective of an external attacker on the WAN. After executing offensive operations, students switched roles to the Blue Team to detect and analyze these activities using firewall logs and Snort IDS.

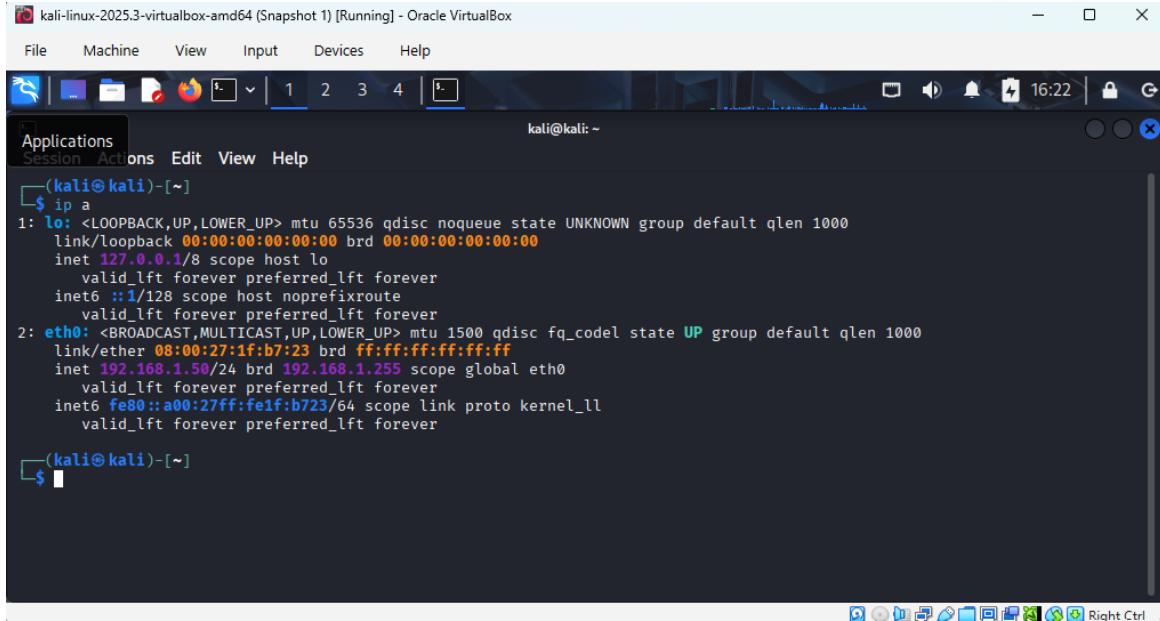
### **Lab Goals**

1. **Preparation:** Configure the open-source firewall to simulate a realistic, semi-restricted DMZ environment.
  2. **Reconnaissance:** Perform port scanning and banner grabbing to identify entry points and service versions.
  3. **Evasion:** Execute SSH tunneling (port forwarding) to bypass firewall rules and access blocked internal services via a compromised DMZ host as a pivot.
  4. **Defense:** Analyze firewall logs and configure Snort on the IAM server to detect intrusive traffic.
-

## 2. Infrastructure Overview

The lab environment consisted of the following topology:

- **Attacker (WAN):** Kali Linux, bridged to student home network, acting as external attacker.

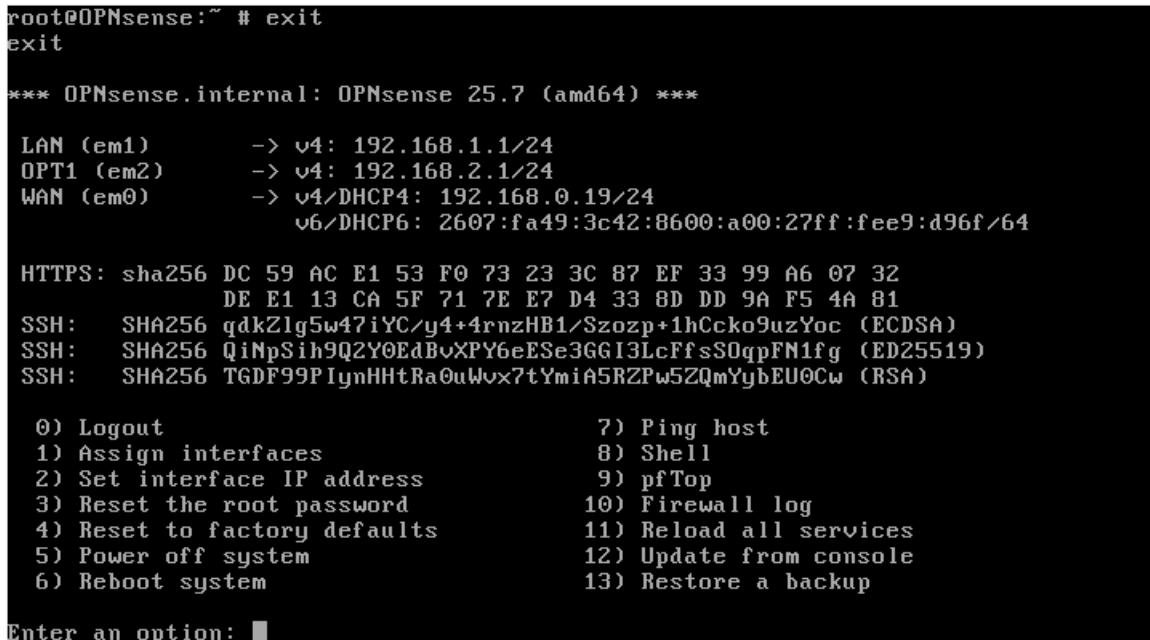


A screenshot of a Kali Linux terminal window titled "kali-linux-2025.3-virtualbox-amd64 (Snapshot 1) [Running] - Oracle VirtualBox". The terminal shows the command \$ ip a and its output. The output shows two interfaces: lo (loopback) and eth0 (ethernet). The lo interface has an IP of 127.0.0.1/8. The eth0 interface has an IP of 192.168.1.50/24 and a MAC address of 08:00:27:1f:b7:23. The terminal prompt is kali@kali: ~

```
(kali㉿kali)-[~]
$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
        inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
            valid_lft forever preferred_lft forever
        inet6 ::1/128 brd :: scope host noprefixroute
            valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:1f:b7:23 brd ff:ff:ff:ff:ff:ff
        inet 192.168.1.50/24 brd 192.168.1.255 scope global eth0
            valid_lft forever preferred_lft forever
        inet6 fe80::a00:27ff:fe1f:b723/64 scope link proto kernel ll
            valid_lft forever preferred_lft forever

(kali㉿kali)-[~]
$
```

- **Firewall:** Open-source firewall (OPNsense) with WAN, LAN, and DMZ interfaces.



A screenshot of an OPNsense terminal session. The user types root@OPNsense:~ # exit and presses enter. The system then displays a configuration screen for the internal interface. It shows three interfaces: LAN (em1), OPT1 (em2), and WAN (em0). The LAN interface has an IP of 192.168.1.1/24. The OPT1 interface has an IP of 192.168.2.1/24. The WAN interface has an IP of 192.168.0.19/24 and a MAC address of 26:07:fa:49:3c:42. The terminal then lists various SSL/TLS certificates. Finally, it presents a menu of 13 options for system management.

```
root@OPNsense:~ # exit
exit

*** OPNsense.internal: OPNsense 25.7 (amd64) ***

LAN (em1)      -> v4: 192.168.1.1/24
OPT1 (em2)     -> v4: 192.168.2.1/24
WAN (em0)      -> v4/DHCP4: 192.168.0.19/24
                  v6/DHCP6: 2607:fa49:3c42:8600:a00:27ff:fee9:d96f/64

HTTPS: sha256 DC 59 AC E1 53 F0 73 23 3C 87 EF 33 99 A6 07 32
      DE E1 13 CA 5F 71 7E E7 D4 33 8D DD 9A F5 4A 81
SSH:   SHA256 qdkZlg5w47iYC/y4+4rnzHB1/Szozp+1hCcko9uzYoc (ECDSA)
SSH:   SHA256 QiNpSih9Q2Y0EdBvXPY6eESe3GGI3LcFFsSOqpFN1fg (ED25519)
SSH:   SHA256 TGDF99P1ynHHtRa0uWvx7tYmiA5RZPw5ZQmYybEU0Cw (RSA)

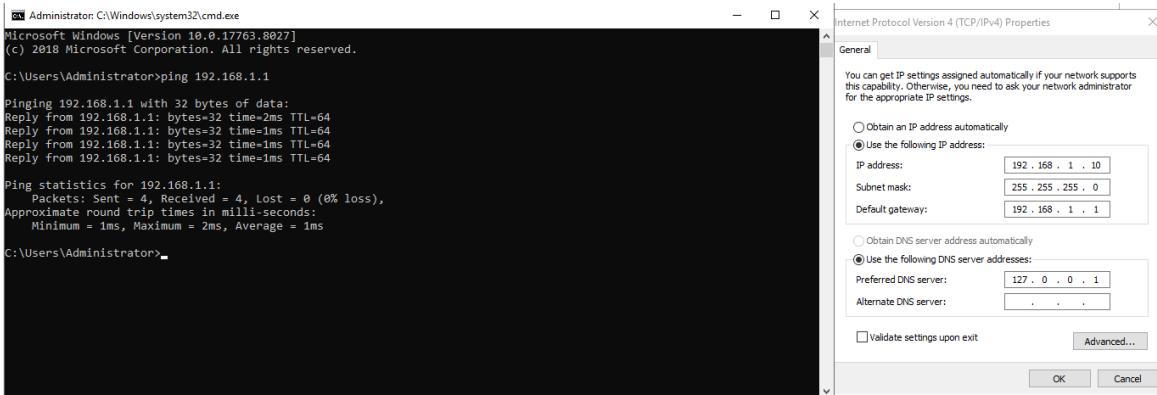
0) Logout          7) Ping host
1) Assign interfaces 8) Shell
2) Set interface IP address 9) pfTop
3) Reset the root password 10) Firewall log
4) Reset to factory defaults 11) Reload all services
5) Power off system 12) Update from console
6) Reboot system 13) Restore a backup

Enter an option: [
```

- **DMZ Target:** Metasploitable / Web Server at IP 192.168.2.10.

```
msfadmin@metasploitable:~$ ip a
1: lo: <LOOPBACK,NOQUEUE,NODISC> mtu 16436 qdisc noop
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast qlen 1000
    link/ether 08:00:27:4e:5e:07 brd ff:ff:ff:ff:ff:ff
    inet 192.168.2.10/24 brd 192.168.2.255 scope global eth0
        inet6 fe80::a00:27ff:fe4e:5e07/64 scope link
            valid_lft forever preferred_lft forever
msfadmin@metasploitable:~$
```

- **LAN Target:** IAM Server (Windows Server 2019 / Active Directory DC) at IP 192.168.1.10, with RDP (TCP 3389) enabled.



This architecture simulates a typical enterprise network where external traffic can only reach selected DMZ services, and direct WAN→LAN access is blocked by firewall policy.

### 3. Phase 0: Firewall Configuration & Baseline

#### 3.1 DMZ Ingress Rules (WAN → DMZ)

**Objective:** Allow public access to web and SSH services on the DMZ while blocking all other traffic.

##### Configuration Steps:

1. Logged into firewall WebGUI.
2. Navigated to: Firewall → Rules → WAN.
3. Created and verified the following rules:

Rule	Action	Protocol	Source	Destination	Port	Purpose
A	Pass	IPv4 TCP	Any	DMZ Net	80	Allow HTTP
B	Pass	IPv4 TCP	Any	DMZ Net	22	Allow SSH
C	Block	IPv4 TCP	Any	DMZ Net	All	Default deny

#### 4. Applied changes and verified active rules in the WebGUI.

The firewall rule configuration has been changed.  
You must apply the changes in order for them to take effect.

**Apply changes**

	Protocol	Source	Port	Destination	Port	Gateway	Schedule	Description	Actions
IPv4 TCP	*	*	DMZ_NET	80 (HTTP)	*	*		Allow HTTP to DMZ	<span style="color: green;">→</span> <span style="color: green;">→</span> <span style="color: green;">⚡</span>
IPv4 TCP	*	*	DMZ_NET	22 (SSH)	*	*		Allow SSH to DMZ	<span style="color: green;">→</span> <span style="color: green;">→</span> <span style="color: green;">⚡</span>
IPv4 *	*	*	*	*	*	*		Block all other traffic	<span style="color: red;">✗</span> <span style="color: red;">→</span> <span style="color: red;">⚡</span>
pass									<span style="color: green;">→</span> <span style="color: green;">in</span>
pass (disabled)									<span style="color: red;">✗</span> <span style="color: red;">→</span> <span style="color: red;">block (disabled)</span>
block									<span style="color: red;">✗</span> <span style="color: red;">reject</span>
block (disabled)									<span style="color: red;">✗</span> <span style="color: red;">reject (disabled)</span>
log									<span style="color: blue;">i</span> <span style="color: blue;">log</span>
log (disabled)									<span style="color: blue;">i</span> <span style="color: blue;">log (disabled)</span>
in									<span style="color: green;">→</span> <span style="color: green;">in</span>
out									<span style="color: red;">✗</span> <span style="color: red;">←</span> <span style="color: red;">out</span>
first match									<span style="color: green;">⚡</span> <span style="color: green;">first match</span>
last match									<span style="color: red;">⚡</span> <span style="color: red;">last match</span>

Active/Inactive Schedule (click to view/edit)  
Alias (click to view/edit)

WAN rules are evaluated on a first-match basis by default (i.e. the action of the first rule to match a packet will be executed). This means that if you use block rules, you will have to pay attention to the rule order. Everything that is not explicitly passed is blocked by default.

### 3.2 LAN Isolation (WAN → LAN Blocked)

**Objective:** Ensure no direct traffic from WAN reaches the internal LAN.

#### Verification Steps:

- Confirmed under Firewall → Rules → WAN that **no** rules permit traffic from WAN to LAN network (192.168.1.0/24).
- From the Attacker machine (Kali), executed tests:  
ping 192.168.1.10  
nmap -sS -Pn 192.168.1.10
- Both ICMP echo and TCP SYN scan timed out or were silently dropped, confirming firewall enforcement.

### 4. Phase A: Port Scanning (Reconnaissance)

#### 4.1 TCP SYN Scan for Open Ports

**Objective:** Identify which TCP ports are reachable on the DMZ server (192.168.2.10).

#### Command (on Kali):

```
nmap -sS -Pn 192.168.2.10
```

#### Parameter Explanation:

- sS: TCP SYN (stealth) scan – does not complete three-way handshake.
- Pn: Assume host is online, skip ICMP ping.

## Results Observed:

Port	State	Service
21/tcp	open	ftp
22/tcp	open	ssh
80/tcp	open	http
445/tcp	open	microsoft-ds
631/tcp	open	ipp
3306/tcp	open	mysql
8080/tcp	open	http-proxy
3000/tcp	closed	ppp
8181/tcp	closed	intermapper

This scan revealed that SSH (port 22) is publicly accessible, making it a viable entry point for tunneling.

```
(kali㉿kali)-[~]
└─$ nmap -sS -Pn 192.168.2.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-14 16:40 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid se
rvers with --dns-servers
Nmap scan report for 192.168.2.10
Host is up (0.0090s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT      STATE SERVICE
21/tcp    open  ftp
22/tcp    open  ssh
80/tcp    open  http
445/tcp   open  microsoft-ds
631/tcp   open  ipp
3000/tcp  closed ppp
3306/tcp  open  mysql
8080/tcp  open  http-proxy
8181/tcp  closed intermapper

Nmap done: 1 IP address (1 host up) scanned in 5.95 seconds
```

## 4.2 Service Version Detection

**Objective:** Determine exact service versions to identify potential vulnerabilities.

### Command:

```
nmap -sV -p 22,80 192.168.2.10
```

### Parameter Explanation:

- **-sV:** Probe open ports to determine service and version information.
- **-p 22,80:** Limit scanning to SSH and HTTP ports for clarity.

## **Results Observed:**

Port	State	Service	Version
22/tcp	open	ssh	OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13
80/tcp	open	http	Apache httpd 2.4.7 (Ubuntu)

**Analysis:** The service versions disclosed indicate Ubuntu Linux as the underlying OS. This information helps an attacker select appropriate exploits or brute-force techniques specific to OpenSSH 6.6.1p1 or Apache 2.4.7.

```
(kali㉿kali)-[~]
└─$ nmap -sV -p 22,80 192.168.2.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-14 16:41 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid se
rvers with --dns-servers
Nmap scan report for 192.168.2.10
Host is up (0.0050s latency).

PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.7 ((Ubuntu))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 6.63 seconds
```

## **4.3 Optional: UDP Port Scan**

### **Command:**

```
nmap -sU --top-ports 20 192.168.2.10
```

**Rationale:** UDP services are sometimes less strictly filtered. This scan probes the 20 most common UDP ports.

---

## **5. Phase B: Banner Grabbing (Service Fingerprinting)**

### **5.1 SSH Banner via Netcat**

**Objective:** Manually retrieve the SSH service banner to confirm version and gather additional system information.

#### **Command (Kali):**

```
nc -v 192.168.2.10 22
```

#### **Expected Output (example):**

```
Connection to 192.168.2.10 22 port [tcp/ssh] succeeded!
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13
```

**Analysis:** The banner string reveals:

- SSH protocol version 2.0
- OpenSSH version 6.6.1p1
- Operating system: Ubuntu (2ubuntu2.13 patch level)

This precise information is valuable for exploit selection and vulnerability assessment.

```
(kali㉿kali)-[~]
$ nc -v 192.168.2.10 22
192.168.2.10: inverse host lookup failed: Host name lookup failure
(UNKNOWN) [192.168.2.10] 22 (ssh) open
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13
```

## 5.2 Automated Banner Grabbing via Nmap NSE

**Command:**

```
nmap -sV --script=banner 192.168.2.10
```

**Purpose:** Nmap's NSE (Nmap Scripting Engine) automates banner retrieval across multiple services and ports simultaneously.

```
(kali㉿kali)-[~]
$ nmap -sV --script=banner 192.168.2.10
Starting Nmap 7.95 ( https://nmap.org ) at 2025-12-14 16:44 EST
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns or specify valid servers with --dns-servers
Nmap scan report for 192.168.2.10
Host is up (0.0080s latency).
Not shown: 991 filtered tcp ports (no-response)
PORT      STATE SERVICE VERSION
21/tcp    open  ftp     ProFTPD 1.3.5
22/tcp    open  ssh     OpenSSH 6.6.1p1 Ubuntu 2ubuntu2.13 (Ubuntu Linux; protocol 2.0)
|_banner: SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13
80/tcp    open  http   Apache httpd/2.4.7 ((Ubuntu))
|_http-server-header: Apache/2.4.7 ((Ubuntu))
445/tcp   open  netbios-ssn Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
631/tcp   open  ipp    CUPS 1.7
|_http-server-header: CUPS/1.7 IPP/2.1
3000/tcp closed ppp
3306/tcp open  mysql   MySQL (unauthorized)
|_banner: E\x00\x00\x00\xFFj\x04Host '192.168.1.50' is not allowed to con
|_nect to this MySQL server
8080/tcp open  http   Jetty 8.1.7.v20120910
|_http-server-header: Jetty/8.1.7.v20120910
8181/tcp closed intermapper
Service Info: Host: METASPLOITABLE3-UB1404; OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
Service detection performed. Please report any incorrect results at https://nmap.org/submit/.
Nmap done: 1 IP address (1 host up) scanned in 36.47 seconds
```

---

## 6. Phase C: SSH Tunneling (Firewall Evasion)

### 6.1 Scenario & Goal

## **Scenario:**

- Firewall allows WAN → DMZ SSH (port 22).
- Firewall blocks WAN → LAN entirely.
- Intelligence (from scanning) confirms an RDP service on LAN at 192.168.1.10:3389.

**Goal:** Use SSH access to DMZ as a jumping-off point (pivot) to tunnel RDP traffic to the internal server.

## **6.2 Establish Local Port Forwarding**

### **Command (Kali - Terminal 1):**

```
ssh -L 3333:192.168.1.10:3389 vagrant@192.168.2.10
```

### **Parameter Breakdown:**

- `-L 3333:192.168.1.10:3389`: Local port forwarding.
  - 3333: Listen port on attacker's Kali machine.
  - 192.168.1.10:3389: Target destination (IAM Server LAN IP : RDP port).
  - `vagrant@192.168.2.10`: SSH user and DMZ jump host.

### **Execution Flow:**

1. SSH connection established to DMZ server.
2. Prompted for password (default for Metasploitable3: vagrant).
3. After authentication, the terminal receives a shell prompt on the DMZ server.
4. The SSH tunnel remains active in the foreground; data sent to `127.0.0.1:3333` on Kali is tunneled through this session to `192.168.1.10:3389` on the LAN.

```
(kali㉿kali)-[~]
$ ssh -L 3333:192.168.1.10:3389 vagrant@192.168.2.10

vagrant@192.168.2.10's password:
Welcome to Ubuntu 14.04 LTS (GNU/Linux 3.13.0-24-generic x86_64)

 * Documentation: https://help.ubuntu.com/
Last login: Sun Dec 14 23:37:42 2025 from 192.168.1.50
vagrant@metasploitable3-ub1404:~$ █
```

## 6.3 Verify Tunnel Establishment

**Command (Kali - Terminal 2):**

```
netstat -antp | grep 3333
```

**Expected Output:**

```
tcp 0 0 127.0.0.1:3333 0.0.0.0:* LISTEN 1234/ssh
```

**Interpretation:**

- Confirms that Kali is listening on port 3333.
- Process owner is ssh, indicating the tunnel is active.
- Traffic on 3333 is bound to 127.0.0.1 (localhost only) and will be forwarded to 192.168.1.10:3389 via the SSH tunnel.



```
kali@kali: ~
Session Actions Edit View Help
[(kali㉿kali)-~]
$ netstat -antp | grep 3333
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
tcp        0      0 127.0.0.1:3333          0.0.0.0:*
              LISTEN      23087/ssh
tcp6       0      0 ::1:3333              ::*:*
              LISTEN      23087/ssh
[(kali㉿kali)-~]
$
```

## 6.4 RDP Connection Over Tunnel

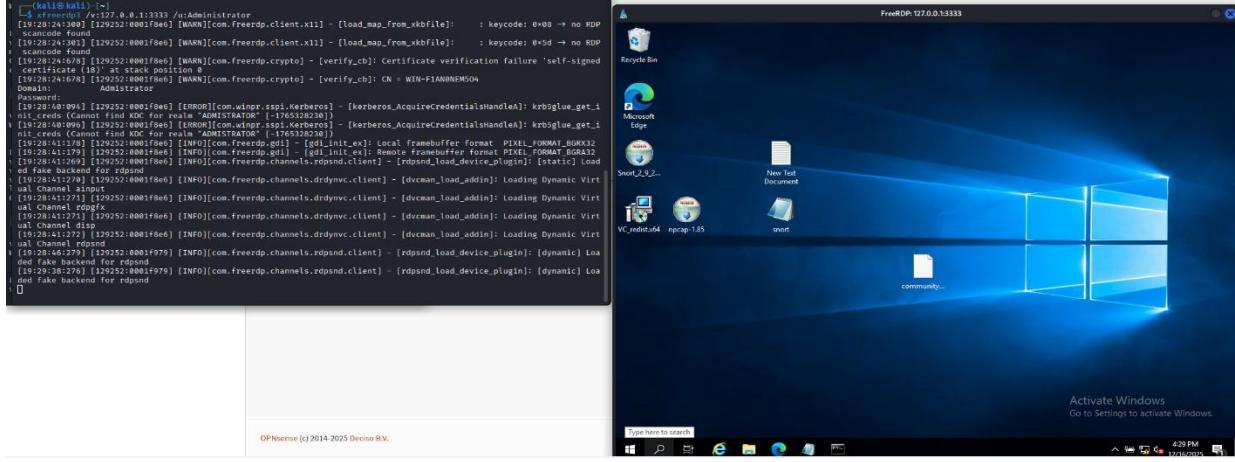
**Command (Kali):**

```
xfreerdp3 /v:127.0.0.1:3333 /u:Administrator
```

**Execution:**

1. RDP client connects to localhost on port 3333.
2. Kali forwards the RDP traffic through the SSH tunnel.
3. DMZ server (192.168.2.10) receives the tunneled RDP traffic and forwards it to IAM Server (192.168.1.10:3389).
4. The firewall observes SSH traffic from WAN to DMZ (allowed).
5. RDP traffic to LAN is encapsulated within SSH and therefore bypasses the WAN→LAN block.

**Result:** The RDP login window appears, showing the IAM Server desktop despite the firewall's WAN→LAN restriction.



The terminal window shows RDP log entries from a Kali Linux host (IP 192.168.0.13333) connecting to a Windows 10 host (IP 192.168.0.13333). The logs detail the negotiation of security protocols (Kerberos, SSL/TLS), the creation of session handles, and the establishment of a remote frame buffer connection. The Windows desktop shows the standard RDP login interface with the Windows logo and a 'command...' button.

```

[!] [root@kali: ~]
[~] $ xtermrdp -v:127.0.0.1:13333 /u:Administrator
[19:28:24:30B] [129252:0:0001f8e6] [WARN][com.freerdp.client.v11] - [load_map_from_xkbfile]: : keycode: 0x08 → no RDP
[19:28:24:30B] [129252:0:0001f8e6] [WARN][com.freerdp.client.v11] - [load_map_from_xkbfile]: : keycode: 0x5d → no RDP
[19:28:24:30B] [129252:0:0001f8e6] [WARN][com.freerdp.crypto] - [verify_cobj] Certificate verification failure 'self-signed'
[19:28:24:30B] [129252:0:0001f8e6] [WARN][com.freerdp.crypto] - [verify_cobj] CN = WIN-TAUNHEMSON4
Domain: Administrator
Password:
[19:28:24:30B] [129252:0:0001f8e6] [INFO][com.wingr.ssl1.Kerberos] - [kerberos_AcquireCredentialsHandle]: krb5glue_get_i
niti_creds (Cannot find KDC for realm 'ADMINISTRATOR' [-1765332828])
[19:28:40:096] [129252:0:0001f8e6] [INFO][com.wingr.ssl1.Kerberos] - [kerberos_AcquireCredentialsHandle]: krb5glue_get_i
niti_creds (Cannot find KDC for realm 'ADMINISTRATOR' [-1765332828])
[19:28:41:178] [129252:0:0001f8e6] [INFO][com.freerdp.gdi] - [gdi_init_ex]: Local framebuffer format PIXEL_FORMAT_BURG32
[19:28:41:178] [129252:0:0001f8e6] [INFO][com.freerdp.gdi] - [gdi_init_ex]: Remote framebuffer format PIXEL_FORMAT_BURG32
[19:28:41:178] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnvnc] - [ropang_load_device_Plugin]: [x11] Load
ed fake backend for rdpnd
[19:28:41:278] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnvnc_client] - [dvcmn_load_admin]: Loading Dynamic Virt
ual Channel: RDPFX
[19:28:41:278] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnvnc_client] - [dvcmn_load_admin]: Loading Dynamic Virt
ual Channel disp
[19:28:41:278] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnvnc_client] - [dvcmn_load_admin]: Loading Dynamic Virt
ual Channel Rpsnd
[19:28:46:279] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnd_client] - [rdpsnd_load_device_plugin]: [dynamic] Loa
ding Dynamic Virtual Channel Rpsnd
[19:29:38:279] [129252:0:0001f8e6] [INFO][com.freerdp.channels.rdpnd_client] - [rdpsnd_load_device_plugin]: [dynamic] Loa
ding Dynamic Virtual Channel Rpsnd
[!] [root@kali: ~]

```

## 7. Phase D: Defense (Blue Team)

### 7.1 Detecting Port Scanning via Firewall Logs

**Objective:** Identify scanning activities in firewall logs.

#### Steps:

1. Logged into firewall WebGUI.
2. Navigated to: Status → System Logs → Firewall → Normal View.
3. Filtered logs by attacker IP address (Kali's WAN IP).
4. Analyzed the log entries.

#### Observations:

During the Nmap SYN scan, the firewall logged numerous **Blocked (Deny)** entries (often displayed as red X symbols) for ports that were not explicitly allowed (i.e., not 80 or 22). These entries correspond to:

- Nmap probing random/sequential ports.
- The firewall silently dropping packets for ports without an allow rule.
- A characteristic "noise signature" of a port scan.

**Interpretation:** Multiple denied attempts from a single source targeting various ports is a classic indicator of reconnaissance activity.

The screenshots show the OPNsense Firewall Log Files: Live View interface. The left screenshot shows a log entry for port 22 (SSH) with source 192.168.1.30:49168 and destination 192.168.2.10:22. The right screenshot shows two log entries: one for port 80 (HTTP) with source 192.168.1.30:44242 and destination 192.168.1.1:443, and another for port 22 (SSH) with source 192.168.1.30:49168 and destination 192.168.2.10:22.

## 7.2 Configuring Snort on IAM Server

**Objective:** Deploy Snort IDS to detect suspicious RDP access from the DMZ.

### 7.2.1 Snort Installation Verification

Confirmed that Snort is installed in C:\Snort on the IAM Server (192.168.1.10).

### 7.2.2 Network Variable Configuration

**File:** C:\Snort\etc\snort.conf

**Configuration Added:**

**Setup the network addresses you are protecting**

ipvar HOME\_NET 192.168.1.10

## Set up the external network addresses

ipvar EXTERNAL\_NET any

### Rationale:

- HOME\_NET: Defined as the IAM Server itself (192.168.1.10).
- EXTERNAL\_NET: Set to "any" to monitor traffic from all sources.

The screenshot shows a Notepad window with a red arrow pointing down from the top section to the bottom section. The top section contains configuration variables like ipvar HOME\_NET and ipvar EXTERNAL\_NET. The bottom section contains various port ranges and paths for rule definitions.

```
# Setup the network addresses you are protecting
ipvar HOME_NET 192.168.1.10
ipvar EXTERNAL_NET any

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network
ipvar SQL_SERVERS $HOME_NET

# List of telnet servers on your network
ipvar TELNET_SERVERS $HOME_NET

# List of ssh servers on your network
ipvar SSH_SERVERS $HOME_NET

# List of ftp servers on your network
ipvar FTP_SERVERS $HOME_NET

# List of sip servers on your network
ipvar SIP_SERVERS $HOME_NET

# List of ports you run web servers on
portvar HTTP_PORTS [80,81,311,383,591,593,981,1228,1414,1741,1830,2381,2881,2889,3897,3128,3702,4343,4848,5250,6988,7080,7081,7144,7145,7510,7777,7779,8080,8088,8091,8012,8028,8085,8088,8090,8110,8123,8180,8181,8243,8288,8300,8888,8899,9000,9050,9090,9091,9443]

# List of ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS 188

# List of ports you might see oracle attacks on
portvar ORACLE_PORTS 1624

# List of ports you want to look for SSH connections on:
portvar SSH_PORTS 22

# List of ports you run Ftp servers on
portvar FTP_PORTS [21,2100,3555]

# List of ports you run SIP servers on
portvar SIP_PORTS [5060,5061,5060]

# List of file data ports for file inspection
portvar FILE_DATA_PORTS [$HTTP_PORTS,110,143]

# List of GTP ports for GTP preprocessor
portvar GTP_PORTS [2123,2152,3386]

# other variables, these should not be modified
ipvar ADM_SERVERS [64.12.24.0/13,64.12.28.0/13,64.12.161.0/24,64.12.163.0/24,64.12.208.0/24,205.188.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.179.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# such as: c:\snort\rules
# or: ./rules
var SS_RULE_PATH ./rules
var PREPROTC_RULE_PATH ./preproc_rules
```

Activate Windows  
Gün für Sattigung in aktuelle Windows

### 7.2.3 Rule Paths Verification

Verified that rule include paths in snort.conf point to C:\Snort\rules for proper rule loading.

## 7.3 Creating a Detection Rule

**Objective:** Write a Snort rule to alert on RDP traffic originating from the DMZ network.

**File:** C:\Snort\rules\local.rules

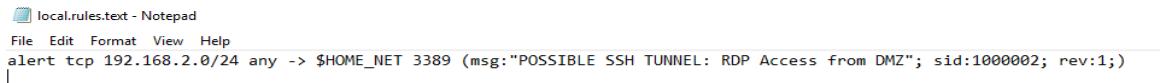
### Rule Added:

```
alert tcp 192.168.2.0/24 any -> $HOME_NET 3389
(msg:"POSSIBLE SSH TUNNEL: RDP Access from DMZ"; sid:1000002; rev:1;)
```

### **Rule Explanation:**

Component	Value	Purpose
alert	—	Generate alert when rule matches
tcp	—	Match TCP protocol
192.168.2.0/24	Source	DMZ network (CIDR notation)
any	Source Port	Any source port
->	Direction	Unidirectional (source → destination)
\$HOME_NET	Destination	IAM Server (192.168.1.10)
3389	Dest Port	RDP protocol port
msg:	—	Human-readable alert message
sid:	1000002	Unique signature ID
rev:	1	Rule revision number

**Security Rationale:** In a secure environment, RDP should only originate from trusted admin subnets, not from a public-facing DMZ server. Any RDP traffic from DMZ to IAM is highly suspicious and likely indicates a compromised pivot point or tunnel.



A screenshot of a Windows Notepad window titled "local.rules.text - Notepad". The window contains a single line of Snort rule syntax: "alert tcp 192.168.2.0/24 any -> \$HOME\_NET 3389 (msg:\"POSSIBLE SSH TUNNEL: RDP Access from DMZ\"; sid:1000002; rev:1;)". The Notepad interface includes standard menu options like File, Edit, Format, View, and Help.

### **7.4 Running Snort and Triggering the Alert**

#### **Snort Execution (on IAM Server - Run as Administrator):**

1. Opened Command Prompt as Administrator.
2. Navigated to Snort binary directory:

```
cd C:\Snort\bin
```

3. Determined the correct interface number:

```
snort -W
```

(Example output: Interface 1 = Ethernet, etc.)

4. Started Snort in IDS mode with console alerting:

```
snort -i 1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii
```

#### **Parameter Explanation:**

- `-i 1`: Listen on interface 1 (LAN adapter).
- `-A console`: Output alerts to console in real-time.
- `-c C:\Snort\etc\snort.conf`: Use the configuration file with our custom rule.
- `-l C:\Snort\log`: Log packets and alerts to this directory.
- `-K ascii`: Log in ASCII format (human-readable).

**Snort Status:** Snort initialized successfully, displayed rules loaded count, and entered packet processing mode.

### **7.5 Triggering and Observing the Alert**

#### **Procedure:**

1. While Snort was running on the IAM Server, returned to the Kali attacker machine.
2. Re-established the SSH tunnel (Terminal 1):  

```
ssh -L 3333:192.168.1.10:3389 vagrant@192.168.2.10
```
3. Initiated RDP connection over the tunnel (Terminal 2):  

```
xfreerdp /v:127.0.0.1:3333 /u:Administrator
```
4. Once RDP session established, observed the Snort console on the IAM Server.

#### **Alert Triggered:**

```
[ ] [1:1000002:1] POSSIBLE SSH TUNNEL: RDP Access from DMZ []
{TCP} 192.168.2.10:XXXXXX -> 192.168.1.10:3389
```

#### **Interpretation:**

- Snort matched the rule condition: TCP traffic from DMZ subnet (192.168.2.10) to HOME\_NET (192.168.1.10) on port 3389.
- Alert flagged the suspicious RDP connection as a potential SSH tunnel attack.
- This demonstrates successful detection of evasion tactics.

## 8. Key Findings & Analysis

### 8.1 Reconnaissance Effectiveness

Port scanning and banner grabbing successfully mapped the target environment:

- Identified open services (SSH, HTTP, etc.).
- Determined exact service versions and OS type.
- Provided sufficient intelligence to select SSH as an entry point.

### 8.2 Evasion Success

SSH local port forwarding proved effective in bypassing firewall restrictions:

- Firewall allowed SSH to DMZ (policy-compliant).
- RDP traffic was encapsulated within SSH (transparent to firewall rules).
- Direct WAN→LAN access remained blocked, but was circumvented via pivot.

### 8.3 Defense Detection

Layered defense mechanisms successfully identified attack activities:

- **Firewall Logs:** Detected reconnaissance noise from port scanning.
- **Snort IDS:** Detected suspicious RDP access pattern inconsistent with normal operations.
- **Combined Analysis:** Traces of attacker activity visible at multiple layers.

### 8.4 Security Lessons

1. **Defense in Depth:** A single firewall rule cannot stop sophisticated attackers; IDS complements perimeter controls.
2. **Pivot Risk:** Compromised DMZ hosts can become pathways to internal networks via tunneling.
3. **Suspicious Pattern Recognition:** RDP from a DMZ web server is inherently anomalous and should trigger alerts.
4. **Monitoring & Logging:** Without logs and IDS rules, the attack would have succeeded silently.

## 9. Recommendations

1. **Network Segmentation:** Implement more granular firewall rules; restrict SSH to specific admin IPs if possible.
  2. **Host-Based Security:** Deploy endpoint detection and response (EDR) on DMZ servers to monitor outbound SSH tunnel activity.
  3. **IDS Tuning:** Expand Snort rules to detect SSH tunneling signatures (anomalous SSH session patterns).
  4. **Access Controls:** Use multi-factor authentication (MFA) on RDP and limit RDP accessibility to VPN or jump servers.
  5. **Incident Response:** Establish procedures to respond to alerts (e.g., disconnect suspicious sessions, isolate hosts).
- 

## 10. Conclusion

This lab demonstrated the complete attack-defense lifecycle:

- **Offensive Phase:** Reconnaissance and evasion techniques enabled an external attacker to achieve unauthorized internal access despite firewall restrictions.
- **Defensive Phase:** Logs and IDS rules provided visibility into attack stages and enabled detection of suspicious activity.

The success of this lab illustrates that security is not achieved through a single control but through **defense in depth**, combining network perimeter security, intrusion detection, log analysis, and incident response procedures. Organizations must continuously monitor for indicators of compromise and maintain updated security rules to defend against evolving threats.

---

## Appendix A: Command Reference

Phase	Command	Purpose
Scanning	<code>nmap -sS -Pn 192.168.2.10</code>	Identify open TCP ports
Scanning	<code>nmap -sV -p 22,80 192.168.2.10</code>	Detect service versions
Banner Grab	<code>nc -v 192.168.2.10 22</code>	Retrieve SSH banner
Evasion	<code>ssh -L 3333:192.168.1.10:3389 vagrant@192.168.2.10</code>	Establish SSH tunnel
Verification	<code>netstat -antp \  grep 3333</code>	Confirm tunnel listening
Evasion	<code>xfreerdp /v:127.0.0.1:3333 /u:Administrator</code>	Access RDP over tunnel
Detection	Firewall: Status → System Logs → Firewall	Review firewall logs
Detection	<code>snort -i 1 -A console -c C:\Snort\etc\snort.conf -l C:\Snort\log -K ascii</code>	Run Snort IDS

---

## **Appendix B: Screenshot Checklist**

- [ ] Screenshot 1: Network Topology Diagram
  - [ ] Screenshot 2: Firewall WAN Rules (A, B, C)
  - [ ] Screenshot 3: Failed ping/Nmap to 192.168.1.10
  - [ ] Screenshot 4: `nmap -sS -Pn 192.168.2.10` output
  - [ ] Screenshot 5: `nmap -sV -p 22,80 192.168.2.10` output
  - [ ] Screenshot 6: UDP scan output (if performed)
  - [ ] Screenshot 7: Netcat SSH banner output
  - [ ] Screenshot 8: `nmap -sV --script=banner` output
  - [ ] Screenshot 9: SSH tunnel established (Terminal 1)
  - [ ] Screenshot 10: netstat LISTEN on port 3333
  - [ ] Screenshot 11: RDP window/Desktop via tunnel
  - [ ] Screenshot 12: Firewall logs showing port scan blocks
  - [ ] Screenshot 13: snort.conf network variable configuration
  - [ ] Screenshot 14: local.rules with RDP detection rule
  - [ ] Screenshot 15: Snort startup and running
-