 ****

Text

Description automatically generatedText

Description automatically generatedText

Description automatically generated

Lab 02 – Firewall Rules & Logs

V1.2

TABLE OF CONTENTS

**[1.](#_Toc170113325)****[Purpose](#_Toc170113325)** [2](#_Toc170113325)

**[2.](#_Toc170113326)****[Prerequisites](#_Toc170113326)** [2](#_Toc170113326)

**[3.](#_Toc170113327)****[Requirements](#_Toc170113327)** [2](#_Toc170113327)

**[4.](#_Toc170113328)****[Instructions](#_Toc170113328)** [2](#_Toc170113328)

# Purpose

In this laboratory, students will set up a pfSense firewall with three network interfaces (WAN, LAN 1, and LAN 2) and configure basic firewall rules to control access between networks. Additionally, students will analyze firewall logs using various command-line tools to extract meaningful data. The purpose of this lab is to reinforce practical skills in firewall configuration, network segmentation, and log analysis.

# Prerequisites

- Basic knowledge of networking concepts (IP addressing, subnets, routing).  
- Understanding of firewall principles and packet filtering.  
- Familiarity with Linux command-line tools.  
- Basic experience with virtualization software.

# Requirements

- A computer with virtualization software (VMware or VirtualBox).  
- pfSense or OpenSense virtual machine.  
- Virtual machines running Kali Linux and Windows.  
- Internet access.

# Instructions

**Activity 1: Configuring a network firewall (40%)**

**Part 1: Configuring the environment (20%)**

Students will create a virtualized environment where PfSense or OpenSense will be the gateway machine for 2 different LAN networks and connect 2 machines with it.

1. Install and boot pfSense or OpenSense in a virtualized environment.  
2. In Firewall Configure three network interfaces:  
 - WAN: Connects to the internet and is used for firewall administration.  
 - LAN 1: Internal network segment (ex: 192.168.1.1/24).  
 - LAN 2: Another internal network segment (ex:192.168.2.1/24).  
3. Connect a Kali Linux machine to LAN 2 interface and a Windows machine to LAN 1 interface.  
4. Disable the Windows firewall to allow connectivity testing.  
5. Test network connectivity using ping.

|  |
| --- |
| LAN 1  WAN  Windows  192.168.1.10  pfSense  LAN 2  Kali  192.168.2.5 |

Example of the network

Provide the following screenshots as evidence of task success:

1.Firewall with 3 interfaces created and the configured interfaces and assigned IP Addresses.

|  |
| --- |
|  |

2. Windows machine with LAN 1 interface as gateway and a local IP in the same CIDR as LAN1.

|  |
| --- |
|  |

3 Kali machine with LAN 2 interface as gateway and local IP in the same CIDR as LAN2.

|  |
| --- |
|  |

4. Windows machine successfully pinging Kali Machine

|  |
| --- |
|  |

5. Kali machine successfully pinging Windows Machine

|  |
| --- |
|  |

**Part 2: Restricting services (20%)**

1.In Windows machines activate RDP service and ensure the local firewall is deactivated.

2. ensure that firewall machines allow traffic to the service (as previously done with ping)

3. From Kali machine connect to the service using any remote tool like freerdp or rdesktop (ex. *rdesktop -u “” <ip\_windows>*)

Provide a screenshot of successful connection from Kali

|  |
| --- |
|  |

4.From the firewall create a rule that blocks “all incoming requests to the RDP service only” other services must stay unrestricted.

Provide a screenshot of the rules applied.

|  |
| --- |
| Screenshot 2025-08-18 192208  Screenshot 2025-08-18 192234 |

5. From Kali show that it is no longer possible to connect to RDP but ping to windows machine still responsible.

Provide a screenshot

|  |
| --- |
|  |

**NOTE:** all the screenshots should be provided clearly showing the requested elements to earn the points.

**Activity 2: Configuring and Restricting Services (30%)**

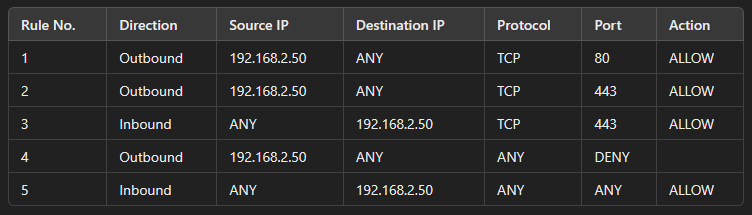
Students will read the following firewall rules and based on the logic and order of them should identify if there is any error, later a proposal of how that error can be solved in the firewall ruleset (ex. Move or delete a rule, correct the grammar etc.).

**Case 1**



|  |
| --- |
| **Describe the error** |
| In the ruleset there is a conflict between Rule 1 and Rule 2. Rule 1 blocks all inbound SSH traffic (port 22) to 192.168.1.10, but Rule 2 should allow SSH only from 192.168.1.100. Since rules are checked in order, Rule 1 takes effect first and Rule 2 never works. This makes Rule 2 useless.Towards the end of Activity 2, when I was blocking RDP access, I encountered this error. I learned that the order of the rules is crucial in firewall configuration, as the first matching rule is applied, and later rules are ignored. |
| **Propose a solution for the error** |
|  |

**Case 2**



|  |
| --- |
| **Describe the error** |
| Rule 4 states that outbound traffic from 192.168.2.50 is denied for all ports and protocols. This rule conflicts with the other rules. Rules 1, 2, and 3 allow certain outbound and inbound traffic for specific ports, like 80 and 443. However, Rule 4 blocks all outbound traffic from 192.168.2.50. This causes unintended blocking of outbound traffic, even though Rules 1 and 2 should permit it. Furthermore, Rule 5 allows inbound traffic for 192.168.2.50. This rule is too broad and could create a security risk since it permits any protocol or port. |
| **Propose a solution for the error** |
| The simplest solution is to adjust Rule 4 so that it is more specific, rather than blocking all outbound traffic. We could rewrite it to block only unnecessary ports or protocols. For example, change Rule 4 to block only non-essential ports, or to allow the necessary ports (80 and 443). Rule 5 should also be more specific to restrict inbound traffic to only the necessary protocols (TCP/443). |

**Case 3**

A screen shot of a black screen

AI-generated content may be incorrect.

|  |
| --- |
| **Describe the error** |
| There is a logical conflict between the inbound and outbound rules. Outbound traffic from 192.168.3.10 to ports 80 (HTTP) and 443 (HTTPS) is allowed by Rules 1 and 2. However, inbound traffic to the same host on those ports is explicitly denied by Rules 3 and 4. This means that the server at 192.168.3.10 can initiate connections to the internet on ports 80 and 443, but it will not accept any inbound responses, effectively breaking normal HTTP/HTTPS communication.Rules 1 and 2 allow outbound traffic from 192.168.3.10 on ports 80 and 443, but Rules 3 and 4 deny inbound traffic on the same ports. Because of this, the server can send requests but it cannot receive the replies, so web traffic will not work correctly. |
| **Propose a solution for the error** |
| The solution is to change Rule 3 and Rule 4 to allow the inbound responses, or remove them completely. Another option is to configure them to only deny new inbound connections but allow established ones, so normal HTTP/HTTPS traffic can work. |

**Activity 3: Log Analysis (30%)**

**Part 1: Understanding log acronyms (10%)**

Understanding what information is saved in the log registries is vital for any kind of investigation or monitoring.

Students should read the following Fortinet log extract and define the meaning (not the value) of each acronym and explain briefly what it means.

|  |
| --- |
| date=2023-05-12 time=15:30:21 logid="0000000013" type="traffic" subtype="forward" level="critical" vd="vdom1" eventtime=1683909021000005000 srcip=10.0.0.10 srcport=8080 srcintf="port3" srcintfrole="undefined" dstip=192.168.2.50 dstport=443 dstintf="port1" dstintfrole="undefined" srcuuid="ae28f494-5735-51e9-f247-d1d2ce663f4b" dstuuid="ae28f494-5735-51e9-f247-d1d2ce663f4b" poluuid="ccb269e0-5735-51e9-a218-a397dd08b7eb" sessionid=10734 proto=6 action="deny" policyid=7 policytype="Antivirus" service="HTTPS" app="Ransomware" appcat="Malware" apprisk="high" applist="g-default" countapp=1 osname="Windows" mastersrcmac="a2:e9:00:ec:40:01" srcmac="a2:e9:00:ec:40:01" utmref=65500-748 |

Fill this table

|  |  |
| --- | --- |
| **Acronym** | **Meaning** |
| date | The date the log was generated |
| time | The exact time when the log entry was generated |
| logid | Unique identifier for the log type/event |
| type | Type of log ( traffic, event) |
| subtype | More detailed classification of the log type |
| level | Severity level |
| vd | Virtual domain (VDOM) where the log event occurred |
| eventtime | Precise event time in epoch/microseconds format (system timestamp) |
| srcip | Source IP address (where the traffic originated from) |
| srcport | Source port number |
| srcintf | Source interface (network interface that sent the traffic) |
| srcintfrole | Role of the source interface (LAN, WAN, undefined) |
| dstip | Destination IP address |
| dstport | Destination port number |
| dstintf | Destination interface.Network interface that receives the traffic |
| dstintfrole | Role of the destination interface |
| srcuuid | Unique identifier (UUID) for the source device/session |
| dstuuid | Unique identifier (UUID) for the destination device/session |
| poluuid | Unique identifier of the applied policy |
| sessionid | ID assigned to the traffic session |
| proto | Protocol used (6 = TCP, 17 = UDP, 1 = ICMP) |
| action | Action taken ( accept, deny, block, reset) |
| policyid | ID of the policy that matched this traffic |
| policytype | Type of policy applied (e.g., Firewall, Antivirus, Web filter) |
| service | Service being used (e.g., HTTPS, HTTP, DNS) |
| app | Detected application (Ransomware, Facebook) |
| appcat | Application category (Malware,Cloud) |
| apprisk | Application risk level (high, medium, low) |
| applist | Application control list and profile used |
| countapp | Number of applications detected in the session |
| osname | Operating system of the source host |
| mastersrcmac | Original source MAC address |
| srcmac | Source MAC address (network layer identifier) |
| utmref | Reference ID for UTM (Unified Threat Management) feature log |

**Part 2: Detecting malicious activity (10%)**

Logs hold information regarding different types of attacks; we just need to read them to understand what happened.

In the zipped file “Fortinet\_logs\_unsolved” exist 9 files with evidence of malicious activities detected:

1. SSH Brute force attack
2. Port Scanning
3. SQL Injection
4. Malware communication
5. Ping scan
6. DDoS
7. IP Spoofing
8. Active exploitation
9. Active ransomware

In the following table indicate which file shows one of the mentioned attacks and what indication makes you think that that is the correct option.

|  |  |  |
| --- | --- | --- |
| **File** | **Malicious activity** | **Justification** |
| Fortinet ex1 | DDoS | policytype="DDoS Protection", app="DDoS Attack", multiple packets and high traffic volume were logged. |
| Fortinet ex2 | Active exploitation | app="MS17-010 Exploit", service="SMB", dstport=445. IPS blocked it. This matches CVE-2017-0144 exploitation. |
| Fortinet ex3 | IP Spoofing | policytype="DoS Protection", app="Spoofed IP Attack". Traffic sent with forged source IP was detected and blocked. |
| Fortinet ex4 | Active ransomware | app="Ransomware", appcat="Malware", policytype="Antivirus", risk marked as high. Fortinet Antivirus blocked it over HTTPS. |
| Fortinet ex5 | Malware communication | app="Trojan", appcat="Malware". Internal host (10.1.2.50) attempted to connect to an external server (203.0.113.45) via HTTPS. Possible C&C connection blocked. |
| Fortinet ex6 | Ping scan | Repeated ICMP probes (subtype="ping", messages like "Machine is up/down") show host discovery reconnaissance. |
| Fortinet ex7 | Port Scanning | The logs show repeated sequential connection attempts from a single source IP (192.168.1.50) to multiple destination ports on 10.10.10.5 (4,6). |
| Fortinet ex8 | SQL Injection | app="SQL.Injection", appcat="Web Application Attack", targeting HTTP (port 80). Blocked by IPS. |
| Fortinet ex9 | SSH Brute force attack | Multiple "Failed login attempt to SSH" entries followed by "Successful login". Typical brute-force credential guessing attack. |

**Part 3: Filtering logs (10%)**

Commercial tools parses and filter log files but this activity can be done manually with basic and local tools to assess the results or as emergency solution when the main tools are not working properly.

Using Linux text manipulation tools (like *grep, awk, sed, wc, and cat)* filter and extract data from the file *log\_example\_1* file and answer the following questions.

|  |  |  |
| --- | --- | --- |
| **#** | **Question** | **Answer** |
| 1 | How many lines does the log file contain? | 56 |
| 2 | How many log entries have the type traffic? | 12 |
| 3 | How many log entries were blocked (action="blocked")? | 10 |
| 4 | How many events have a severity level of critical? | 2 |
| 5 | How many SSH access attempts were blocked? | 0 |
| 6 | What is the most frequent source IP address in the logs? | 10.1.100.22 |
| 7 | How many log entries contain the word virus? | 1 |
| 8 | What is the name of the detected malicious file? | eicar.com |
| 9 | How many HTTPS connection attempts were blocked? | 5 |
| 10 | What are the unique IP addresses appearing in the logs? | 91.189.89.223  50.1.1.101  50.1.1.100  31.1.1.1  23.59.154.35  230.1.1.2  208.91.114.4  208.91.113.83  172.18.70.15  172.16.200.171  172.16.200.11  172.16.200.1  104.80.88.154 |
| 11 | How many log entries contain the action deny? | 0 |
| 12 | How many log entries correspond to detected attacks (attack=)? | 2 |
| 13 | How many connections were accepted (action="accept")? | 3 |
| 15 | How many log entries correspond to failed authentication attempts? | 0 |

**NOTE FOR ADVANCED STUDENTS ONLY**

As you saw it’s possible to create a command to extract precise information from a log file, can you create a script to automate this process?

import re

import argparse

import pandas as pd

def parse\_log\_line(line):

log\_entry = {}

matches = re.findall(r'(\w+)=("[^"]\*"|\S+)', line)

for key, value in matches:

log\_entry[key] = value.strip('"')

return log\_entry

def filter\_logs(logs, filters):

filtered = []

for log in logs:

if all(log.get(k) == v for k, v in filters.items()):

filtered.append(log)

return filtered

def main():

parser = argparse.ArgumentParser(description="FortiGate Log Parser")

parser.add\_argument("logfile", nargs="?", default="Fortinet.txt",

help="Path to the log file (default: Fortinet.txt)")

parser.add\_argument("-f", "--filter", nargs="\*", help="Filter in key=value format (e.g., type=event user=admin)")

parser.add\_argument("-o", "--output", default="parsed\_logs.xlsx", help="Output Excel file name (default: parsed\_logs.xlsx)")

args = parser.parse\_args()

filters = {}

if args.filter:

for f in args.filter:

if "=" in f:

k, v = f.split("=", 1)

filters[k] = v

logs = []

try:

with open(args.logfile, "r") as f:

for line in f:

log\_entry = parse\_log\_line(line)

if log\_entry:

logs.append(log\_entry)

except FileNotFoundError:

print(f"[!] Log file '{args.logfile}' not found.")

return

if filters:

logs = filter\_logs(logs, filters)

if logs:

df = pd.DataFrame(logs)

df.to\_excel(args.output, index=False, engine="openpyxl")

print(f"[+] Extracted {len(logs)} logs saved to {args.output}")

else:

print("[!] No logs matched the filter.")

if \_\_name\_\_ == "\_\_main\_\_":

main()