ÇUKUROVA UNIVERSITY

ENGINEERING AND ARCHITECTURE FACULTY

DEPARTMAN OF COMPUTER ENGINEERING

GRADUATION THESIS

SUBJECT

" SemiGuard: An IDS-Based Desktop Application for Detecting Network Attacks In Virtualized Environments "

By

Semih Zenginoğlu

2025555072

ADVISOR

Doç. Dr. Fatih ABUT

JUNE 2025

ADANA

1.INTRODUCTION

In recent years, the widespread use of virtualization technologies has made network simulations an essential tool, especially for educational and testing purposes. These environments allow the creation of complex networks without physical hardware, enabling safe experimentation with penetration testing and attack detection methods.

This graduation project introduces a desktop-based Intrusion Detection System (IDS) application named **SemiGuard**, developed and tested within a GNS3 and VMware-based simulated network. The simulation includes a Windows virtual machine as the victim, Kali Linux as the attacker, and Ubuntu as the detection unit, all connected via a NAT-based virtual switch.

The Ubuntu machine runs custom scripts that detect four major attacks: **ARP Poisoning**, **DDoS**, **ICMP Flooding**, and **Brute-force** attempts. These scripts are embedded in a user-friendly desktop application that automatically executes them, monitors the network in real time, and displays alerts via a graphical interface.

Each detected attack is logged separately for later analysis. Additionally, the system sends real-time email alerts through Gmail, allowing quick administrative response.

Rather than aiming for enterprise deployment, SemiGuard is designed as a simple, modular, and educational IDS solution—ideal for students and researchers seeking practical experience in network security and intrusion detection.

2. LITERATURE REVIEW

Research in the field of network security has increasingly focused on developing techniques for detecting cyberattacks within simulation environments. Tools such as GNS3 allow realistic network topologies to be emulated without physical infrastructure, enabling the safe testing of attack scenarios and the deployment of intrusion detection systems (IDS). This section examines recent studies that directly relate to the methodologies

employed in the development of the SemiGuard system.

2.1. ARP Spoofing and Flooding Attacks

In a study conducted using GNS3, ARP-based attacks including ARP Flooding, ARP Spoofing, and ARP Poisoning were simulated and detected using a third-party tool named XArp **[1]**. Tools such as Nping, Arpspoof, and Ettercap were utilized to execute these attacks, and the resulting network behavior was analyzed.

SemiGuard adopts a similar simulation-based approach to ARP Spoofing detection within the GNS3 environment. However, rather than relying on external tools like XArp, it employs a custom Python script that continuously monitors the ARP table for anomalies. When a spoofed MAC address is identified, the system logs the incident and sends an alert via email.

2.2. DDoS and ICMP Flooding Detection

Another study simulated DDoS attacks against an HTTP server using GNS3 and evaluated the impact of the attack on system performance **[2]**. The findings highlighted the effectiveness of simulation environments in emulating real-world attack traffic for testing network-based defense mechanisms.

SemiGuard implements a similar approach by using the Scapy library to detect ICMP Flooding attacks. It monitors the number of ICMP packets received within a defined time interval, and if the number exceeds a predefined threshold, the incident is logged and an email alert is dispatched.

2.3. TCP SYN Flood and Brute-Force Detection

TCP SYN Flood attacks are known to overwhelm system resources and cause service disruptions. In one study, a method based on TCP payload analysis was proposed for detecting such attacks **[3]**.

SemiGuard employs a real-time packet sniffing mechanism using Scapy to monitor SYN-flagged TCP packets. If a high volume of such packets is detected from a single IP within a short time frame, a potential SYN flood attack is flagged. Additionally, the tool uses `tshark` to monitor HTTP POST requests and identify brute-force login attempts based on repetitive request patterns from specific IP addresses.

2.4. Dynamic IDS Systems in Virtual Environments

Research on IDS systems deployed within virtualized infrastructures has emphasized the need for adaptability to different attack patterns. The performance of such systems often depends on the network structure and the nature of the threats **[4]**.

SemiGuard, while operating in a virtual simulation, focuses on basic attack types and employs fixed thresholds within modular Python scripts. This makes it suitable for educational and experimental use cases rather than dynamic production environments.

2.5. HTTP POST Flood Attacks and Educational Security Labs

Several projects developed for cybersecurity education have concentrated on HTTP POST flood and brute-force

The SemiGuard intrusion detection system developed in this study was designed and tested within a virtual network environment. The simulation infrastructure was created using an integrated setup of GNS3 (Graphical Network Simulator-3) and VMware Workstation. The system architecture consists of three virtual machines (VMs) with distinct roles: a detection unit (Administrator) running on Ubuntu, an attacker machine (Intruder) based on Kali Linux, and a victim machine (Victim) running Windows 10.

attacks **[5]**. These implementations are intended to support both the theoretical and practical aspects of network security training.

SemiGuard aligns with this objective by offering a GUI-based desktop application capable of real-time traffic monitoring, alert notifications, and separate log generation for each attack type. These features make it a practical tool for hands-on learning in network security laboratories.

3. SYSTEM DESIGN AND ARCHITECTURE

3.1 Network Topology

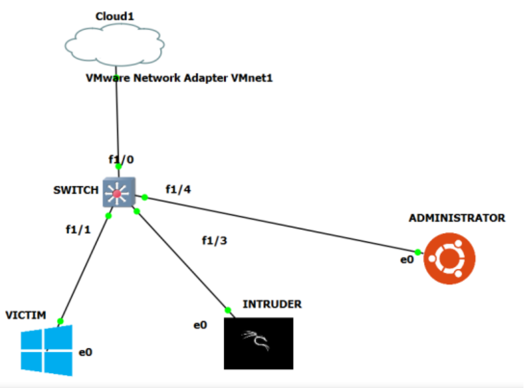
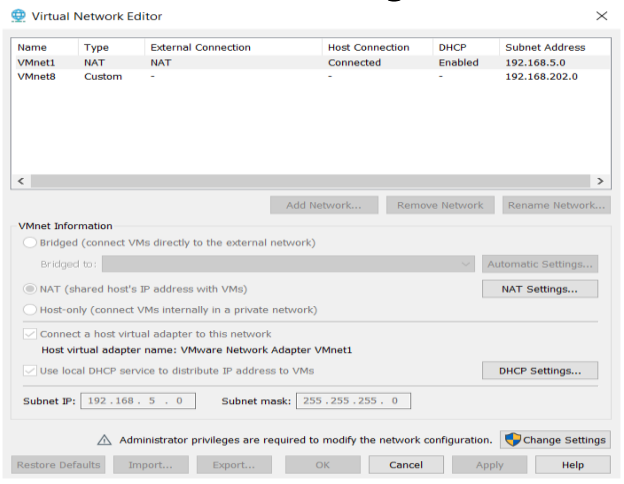


Figure1– Network Topology

The overall network topology of the system is illustrated in Figure 1. It consists of a cloud node representing the NAT-based internet connection, a virtual Ethernet switch (EtherSwitch), and three

VMware-based virtual machines, each connected via dedicated Ethernet ports within the GNS3 simulation platform. The cloud node enables external network access through VMware's VMnet1 adapter, while all devices operate within the same 192.168.5.0/24 subnet to facilitate full-layer communication and packet visibility. The virtual machines serve distinct roles in the simulation: the Windows 10 machine functions as the victim system targeted by various cyberattacks; the Kali Linux machine operates as the attacker, utilizing tools such as Nmap, Ettercap, and Hydra to generate malicious traffic; and the Ubuntu 64-bit machine acts as the administrator node, running the SemiGuard desktop application and executing custom Python-based detection scripts to monitor, analyze, and report suspicious network behavior.

3.2 VMnet1 NAT Configuration


Virtual Network Editor

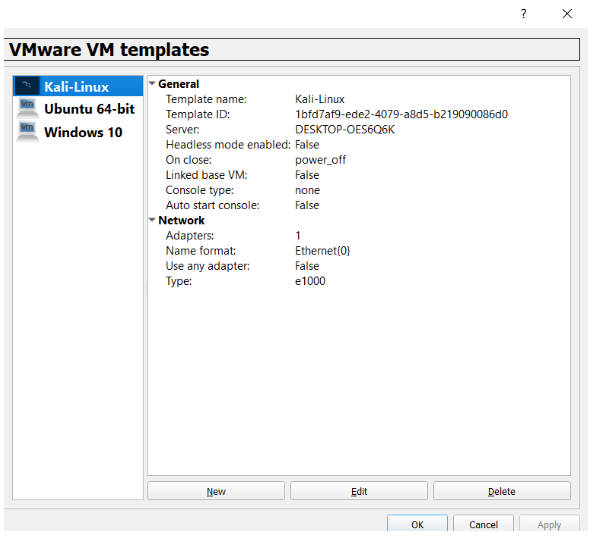
Name	Type	External Connection	Host Connection	DHCP	Subnet Address
VMnet1	NAT	NAT	Connected	Enabled	192.168.5.0
VMnet8	Custom	-	-	-	192.168.202.0

VMnet Information
Bridged (connect VMs directly to the external network)
Bridged to:
[✓ NAT (shared host's IP address with VMs)]
Host-only (connect VMs internally in a private network)
✓ Connect a host virtual adapter to this network
Host virtual adapter name: VMware Network Adapter VMnet1
✓ Use local DHCP service to distribute IP address to VMs
Subnet IP: 192 . 168 . 5 . 0 Subnet mask: 255 . 255 . 255 . 0
Administrator privileges are required to modify the network configuration.

Figure 2– VMware VMnet1 NAT network configuration

The communication between GNS3 and VMware Workstation is established via the VMnet1 virtual adapter, which operates in NAT mode. This adapter provides IP addresses dynamically using DHCP within the 192.168.5.0/24 range and allows virtual machines to communicate with each other as well as access the internet forupdates and SMTP functionality. The configuration details of VMnet1 are shown in Figure 2.

3.3 Virtual Machine Integration


VMware VM templates

Kali-Linux
Ubuntu 64-bit
Windows 10

General

Template name:	Kali-Linux
Template ID:	1bfd7af9-ede2-4079-a8d5-b219090086d0
Server:	DESKTOP-OES6Q6K
Headless mode enabled:	False
On close:	power_off
Linked base VM:	False
Console type:	none
Auto start console:	False

Network

Adapters:	1
Name format:	Ethernet{0}
Use any adapter:	False
Type:	e1000

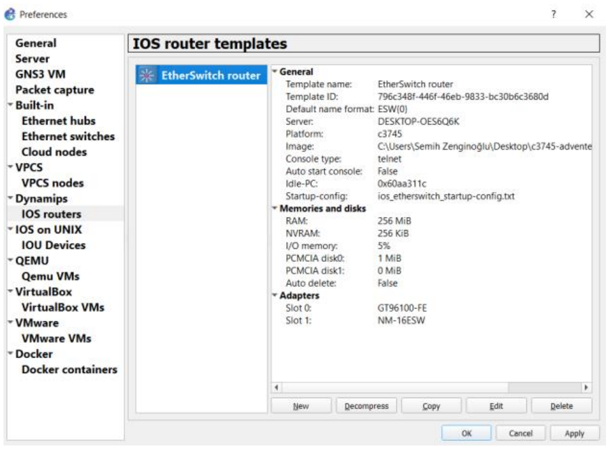
Figüre 3 – VM's integration of the GNS3

Three separate VMware-based virtual machines—Kali Linux, Ubuntu, and Windows 10—were imported into GNS3 through the "VMware VMs" interface.

Each machine is configured with a single Ethernet0 adapter and linked to the GNS3 switch. As shown in Figure 3, these VMs are properly configured to operate within

the same network segment and respond to DHCP.

3.4 Virtual Switch Configuration



```
General
    Template name:        EtherSwitch router
    Template ID:          796c348f-446f-46eb-9833-bc30b6c3680d
    Default name format:  ESW{0}
    Server:               DESKTOP-OES6Q6K
    Platform:             c3745
    Image:                C:\Users\Semih Zenginoğlu\Desktop\c3745-advente
    Console type:         telnet
    Auto start console:   False
    Idle-PC:              0x60aa311c
    Startup-config:       ios_etherswitch_startup-config.txt
Memories and disks
    RAM:                  256 MiB
    NVRAM:                256 KiB
    I/O memory:           5%
    PCMCIA disk0:         1 MiB
    PCMCIA disk1:         0 MiB
    Auto delete:          False
Adapters
    Slot 0:               GT96100-FE
    Slot 1:               NM-16ESW
```

Figure 4 – Configuration panel of the virtual EtherSwitch router in GNS

All virtual machines are connected through a Cisco IOS-based virtual switch configured within GNS3. This switch uses a c3745 router image with the NM-16ESW module to simulate a fully functional Ethernet switch. As shown in Figure 4, this EtherSwitch router replicates physical switching behavior and allows seamless Layer 2 communication between the nodes.

This setup ensures that the detection system can observe all network traffic within the simulation. The Cloud node also facilitates limited internet access, which is essential for system updates, package installations, and email-based attack notifications via SMTP.

4. METHODOLOGY

This section outlines the methodological framework employed in the development of the SemiGuard intrusion detection system. The system was designed with a modular architecture, enabling the identification of four fundamental types of network-based attacks: ARP spoofing, ICMP flooding, TCP SYN flooding (a variant of DDoS), and brute-force login attempts. Each detection mechanism was implemented as a standalone Python script and collectively integrated into a PyQt5-based graphical user interface (GUI) for centralized execution and monitoring.

The methodology involved the systematic development of rule-based detection algorithms for each attack type, guided by a clear objective to maintain simplicity, interpretability, and educational applicability. The detection scripts were tailored for use within a virtualized network environment constructed via

GNS3 and VMware, simulating a real-world topology comprising an attacker machine, a target victim, and a monitoring administrator node.

4.1 ARP Spoofing Detection

The ARP spoofing detection module periodically scans the system’s ARP table to verify IP-to-MAC address consistency. A trusted baseline of ARP entries is maintained, and each new scan is compared against this reference. If a discrepancy is identified—indicating that the MAC address associated with a given IP has changed unexpectedly—the system flags the entry as suspicious. In such cases, the compromised ARP record is programmatically removed from the table, and a log entry is created alongside the dispatch of an email alert to the administrator.

4.2 ICMP Flood Detection

The ICMP flood detection script listens for ICMP Echo Request packets and analyzes their frequency per source IP address. If the number of packets received from a single IP exceeds a predefined threshold within a short time interval (e.g., one second), the behavior is classified as an ICMP flooding attempt. A notification is immediately generated, and the incident is logged. To prevent repeated notifications for ongoing attacks, a cooldown mechanism is employed, ensuring that alerts from the same IP are rate-limited over a defined period.

4.3 TCP SYN Flood (DDoS) Detection

To detect TCP SYN flood attacks, the system monitors all incoming SYN-flagged packets. The script counts the total number of SYN packets received over a specified time window and evaluates whether the rate surpasses the defined threshold. Upon detection, the source IPs with the highest volume of SYN packets are identified and recorded. An email alert is simultaneously triggered to inform the user of the potential denial-of-service attempt, and the relevant details are archived in a dedicated log file.

4.4 Brute-Force Attack Detection

The brute-force detection mechanism uses a dual-layered approach. First, it employs tshark to passively monitor HTTP POST requests, which are often indicative of login attempts. If an abnormal number of POST requests are detected from a specific IP within a limited duration, it is interpreted as a brute-force attack. Second, the system also inspects TCP SYN connection patterns to detect rapid connection attempts, which may reflect automated login guessing. Both layers of detection produce logs and trigger email notifications in case of suspected attack behavior.

4.5 Integration via Graphical User Interface

All four detection scripts are orchestrated through a desktop-based GUI developed using PyQt5. Upon activation by the user, each script is executed as an independent subprocess, allowing for concurrent real-time monitoring. Detected anomalies are displayed in a console-style output window within the interface, providing visibility into the system's operational status. The GUI was designed with accessibility and clarity in mind, adopting a light orange color scheme and a minimal layout to accommodate users with varying technical expertise.

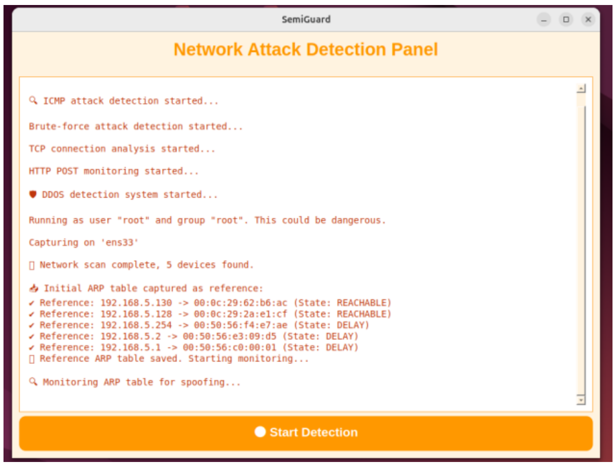
Overall, this methodology facilitates effective detection of common network-layer attacks in a simulated environment, while offering an educational platform for understanding intrusion detection techniques. The modular design allows for future extension with additional detection logic or integration with external data sources and response systems.

The complete source code of all detection scripts and GUI implementation is publicly available on GitHub: https://github.com/semihzen/SemiGuard-Graduation-Thesis-

5.TESTING AND RESULTS

In this section, the functionality of the developed SemiGuard application is evaluated through real-time attack scenarios. For each type of attack, manual intrusions were carried out against target systems, the detection processes were observed, and the accuracy of the resulting outputs was assessed. During the tests, the outputs displayed on the graphical user interface, the contents of the generated log files, and the email notifications sent to the user were individually analyzed and evaluated.

Figure 5 - SemiGuard application shortcut and log directory on desktop



```
🔍 ICMP attack detection started...

Brute-force attack detection started...

TCP connection analysis started...

HTTP POST monitoring started...

🛡 DDOS detection system started...

Running as user "root" and group "root". This could be dangerous.

Capturing on 'ens33'

▯ Network scan complete, 5 devices found.

📥 Initial ARP table captured as reference:
✔ Reference: 192.168.5.130 -> 00:0c:29:62:b6:ac (State: REACHABLE)
✔ Reference: 192.168.5.128 -> 00:0c:29:2a:e1:cf (State: REACHABLE)
✔ Reference: 192.168.5.254 -> 00:50:56:f4:e7:ae (State: DELAY)
✔ Reference: 192.168.5.2 -> 00:50:56:e3:09:d5 (State: DELAY)
✔ Reference: 192.168.5.1 -> 00:50:56:c0:00:01 (State: DELAY)
▯ Reference ARP table saved. Starting monitoring...

🔍 Monitoring ARP table for spoofing...
```

Figure 6 -SemiGuard interface: application startup and detection initialization screen

5.1 Application Initialization and Interface Overview

SemiGuard is launched on an Ubuntu-based virtual machine via a desktop shortcut. As shown in Figure 5, the application resides alongside a dedicated "log" directory on the desktop. This directory stores separate log files for each type of detected attack, maintaining time-stamped records of all security events for retrospective analysis.

Upon execution, the user is presented with a graphical user interface (GUI) designed for simplicity and clarity (Figure 6). When the "Start Detection" button is clicked, the system initiates four independent Python-based detection scripts in parallel: ARP Spoofing, ICMP Flooding, TCP SYN Flood (DDoS), and Brute-force Attack detection modules. The log outputs from each detection routine are displayed in real time within the interface, allowing the user to monitor system behavior continuously.

In addition, the central text console in the GUI provides detailed updates on the status of detection processes. For instance, it displays notifications regarding the capture of the initial ARP table, identification of active network devices, and the initialization of each specific detection script. These updates serve as visual confirmation that the system has been successfully launched and is actively monitoring the network for malicious behavior.

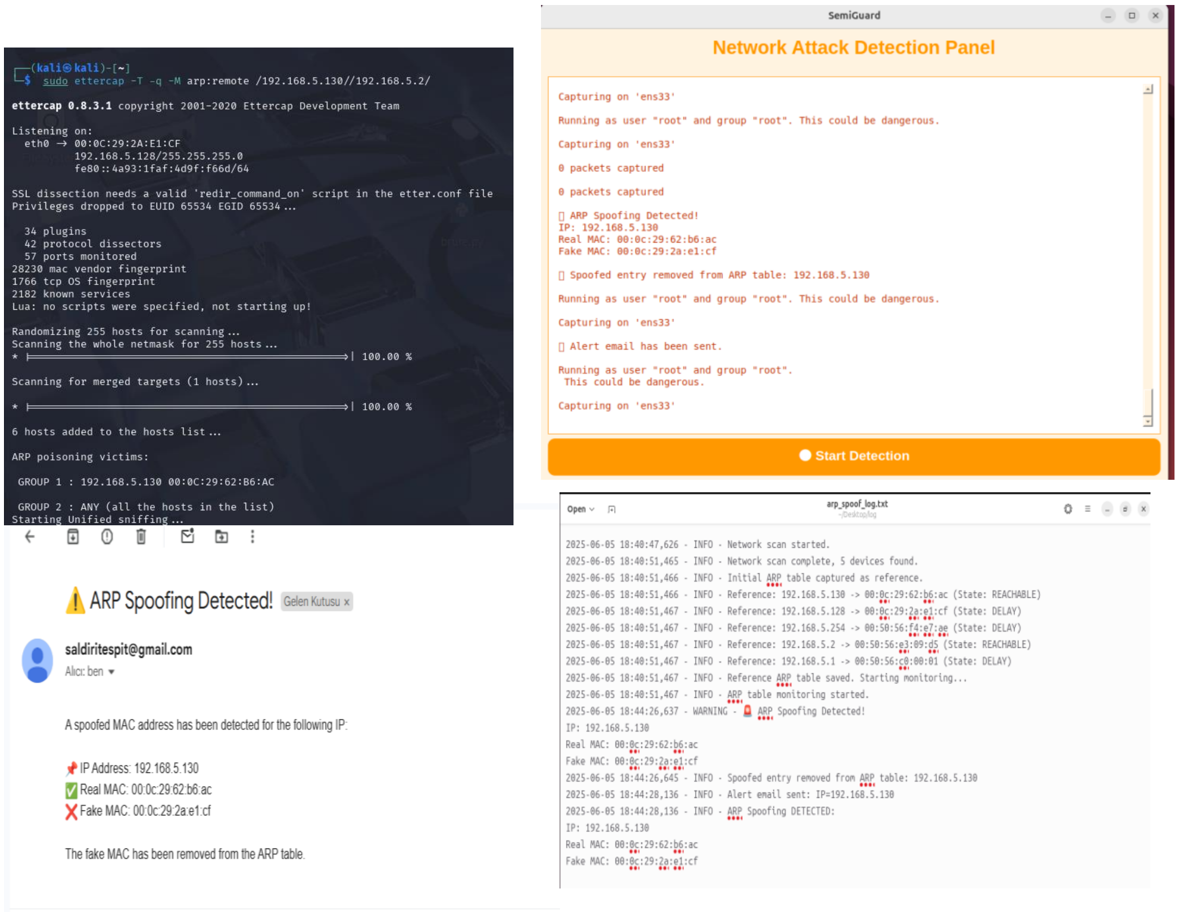


Figure 7 -ARP Spoofing attack execution, detection, logging, and alert email via SemiGuard.

```
┌──(kali㉿kali)-[~]
└─$ sudo ettercap -T -q -M arp:remote /192.168.5.130//192.168.5.2/

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
  eth0 → 00:0C:29:2A:E1:CF
          192.168.5.128/255.255.255.0
          fe80::4a93:1faf:4d9f:f66d/64

SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534 ...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning ...
Scanning the whole netmask for 255 hosts ...
* |==================================================>| 100.00 %

Scanning for merged targets (1 hosts) ...

* |==================================================>| 100.00 %

6 hosts added to the hosts list ...

ARP poisoning victims:

 GROUP 1 : 192.168.5.130 00:0C:29:62:B6:AC

 GROUP 2 : ANY (all the hosts in the list)
Starting Unified sniffing ...
```

SemiGuard

Network Attack Detection Panel

```
Capturing on 'ens33'

Running as user "root" and group "root". This could be dangerous.

Capturing on 'ens33'

0 packets captured

0 packets captured

▯ ARP Spoofing Detected!
IP: 192.168.5.130
Real MAC: 00:0c:29:62:b6:ac
Fake MAC: 00:0c:29:2a:e1:cf

▯ Spoofed entry removed from ARP table: 192.168.5.130

Running as user "root" and group "root". This could be dangerous.

Capturing on 'ens33'

▯ Alert email has been sent.

Running as user "root" and group "root".
 This could be dangerous.

Capturing on 'ens33'
```

⚠️ ARP Spoofing Detected! Gelen Kutusu ×

saldiritespit@gmail.com
Alıcı: ben

A spoofed MAC address has been detected for the following IP:

📌 IP Address: 192.168.5.130
✅ Real MAC: 00:0c:29:62:b6:ac
❌ Fake MAC: 00:0c:29:2a:e1:cf

The fake MAC has been removed from the ARP table.

arp_spoof_log.txt
~/Desktop/log

```
2025-06-05 18:40:47,626 - INFO - Network scan started.
2025-06-05 18:40:51,465 - INFO - Network scan complete, 5 devices found.
2025-06-05 18:40:51,466 - INFO - Initial ARP table captured as reference.
2025-06-05 18:40:51,466 - INFO - Reference: 192.168.5.130 -> 00:0c:29:62:b6:ac (State: REACHABLE)
2025-06-05 18:40:51,467 - INFO - Reference: 192.168.5.128 -> 00:0c:29:2a:e1:cf (State: DELAY)
2025-06-05 18:40:51,467 - INFO - Reference: 192.168.5.254 -> 00:50:56:f4:e7:ae (State: DELAY)
2025-06-05 18:40:51,467 - INFO - Reference: 192.168.5.2 -> 00:50:56:e3:09:d5 (State: REACHABLE)
2025-06-05 18:40:51,467 - INFO - Reference: 192.168.5.1 -> 00:50:56:c0:00:01 (State: DELAY)
2025-06-05 18:40:51,467 - INFO - Reference ARP table saved. Starting monitoring...
2025-06-05 18:40:51,467 - INFO - ARP table monitoring started.
2025-06-05 18:44:26,637 - WARNING - 🚨 ARP Spoofing Detected!
IP: 192.168.5.130
Real MAC: 00:0c:29:62:b6:ac
Fake MAC: 00:0c:29:2a:e1:cf
2025-06-05 18:44:26,645 - INFO - Spoofed entry removed from ARP table: 192.168.5.130
2025-06-05 18:44:28,136 - INFO - Alert email sent: IP=192.168.5.130
2025-06-05 18:44:28,136 - INFO - ARP Spoofing DETECTED:
IP: 192.168.5.130
Real MAC: 00:0c:29:62:b6:ac
Fake MAC: 00:0c:29:2a:e1:cf
```

5.2 ARP Spoofing Attack Test

In this test, a spoofing attack was conducted using a Kali Linux virtual machine and the Ettercap tool to target communication between a victim **(192.168.5.130)** and the gateway **(192.168.5.2).** The attack was initiated via the following command:

sudo ettercap -T -q -M arp:remote /192.168.5.130//192.168.5.2/

Ettercap output confirmed that the attacker injected forged ARP responses, altering the victim's ARP table.

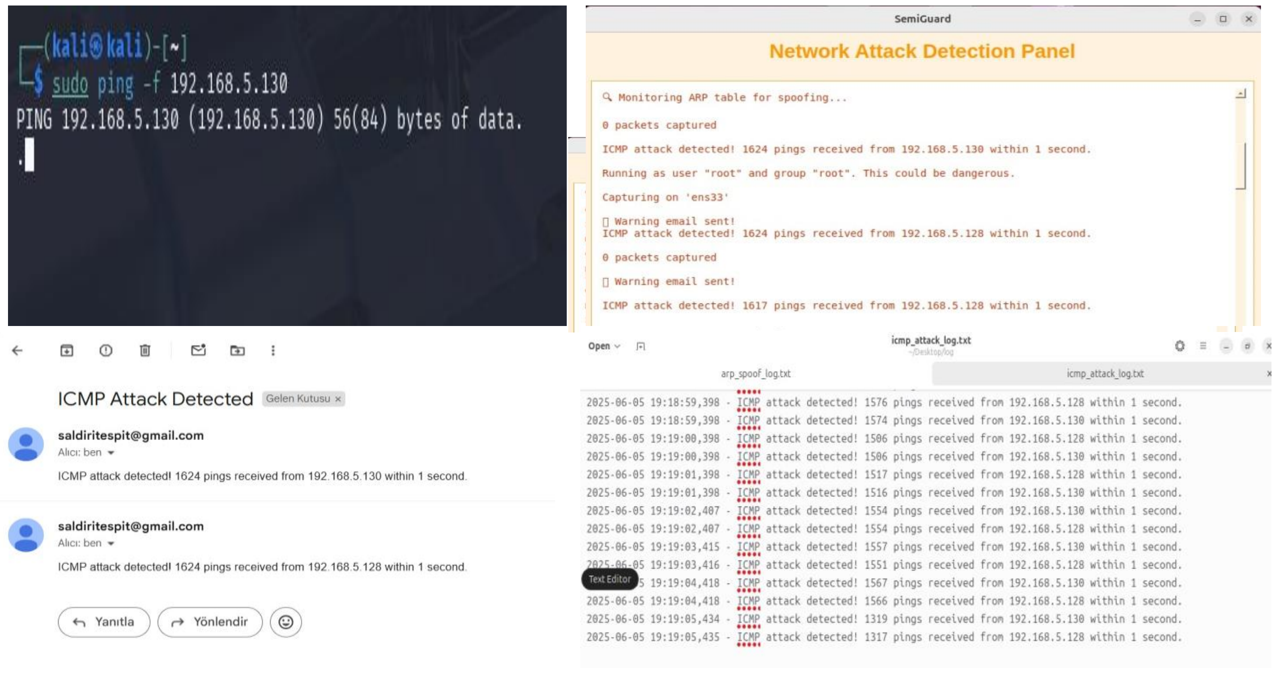
SemiGuard successfully detected the anomaly by comparing the live ARP table with its reference state. The GUI displayed a warning showing both the legitimate **(00:0c:29:62:b6:ac)** and spoofed **(00:0c:29:2a:e1:cf)** MAC addresses for the same IP

As shown in Figure 7, the system:

I. Logged the incident in arp_spoof_log.txt
II. Removed the spoofed ARP entry
III. Sent an alert email to the administrator with full attack details

This test demonstrated SemiGuard's capability not only to detect ARP Spoofing but also to autonomously take corrective action to maintain network integrity.

Figure 8 – Execution and detection of ICMP Flooding attack with log and email alert



```
┌──(kali㉿kali)-[~]
└─$ sudo ping -f 192.168.5.130
PING 192.168.5.130 (192.168.5.130) 56(84) bytes of data.
.
```

SemiGuard

Network Attack Detection Panel

```
🔍 Monitoring ARP table for spoofing...

0 packets captured

ICMP attack detected! 1624 pings received from 192.168.5.130 within 1 second.

Running as user "root" and group "root". This could be dangerous.

Capturing on 'ens33'

▯ Warning email sent!
ICMP attack detected! 1624 pings received from 192.168.5.128 within 1 second.

0 packets captured

▯ Warning email sent!

ICMP attack detected! 1617 pings received from 192.168.5.128 within 1 second.
```

ICMP Attack Detected Gelen Kutusu ×

saldiritespit@gmail.com
Alıcı: ben
ICMP attack detected! 1624 pings received from 192.168.5.130 within 1 second.

saldiritespit@gmail.com
Alıcı: ben
ICMP attack detected! 1624 pings received from 192.168.5.128 within 1 second.

```
2025-06-05 19:18:59,398 - ICMP attack detected! 1576 pings received from 192.168.5.128 within 1 second.
2025-06-05 19:18:59,398 - ICMP attack detected! 1574 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:00,398 - ICMP attack detected! 1506 pings received from 192.168.5.128 within 1 second.
2025-06-05 19:19:00,398 - ICMP attack detected! 1506 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:01,398 - ICMP attack detected! 1517 pings received from 192.168.5.128 within 1 second.
2025-06-05 19:19:01,398 - ICMP attack detected! 1516 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:02,407 - ICMP attack detected! 1554 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:02,407 - ICMP attack detected! 1554 pings received from 192.168.5.128 within 1 second.
2025-06-05 19:19:03,415 - ICMP attack detected! 1557 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:03,416 - ICMP attack detected! 1551 pings received from 192.168.5.128 within 1 second.
          5 19:19:04,418 - ICMP attack detected! 1567 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:04,418 - ICMP attack detected! 1566 pings received from 192.168.5.128 within 1 second.
2025-06-05 19:19:05,434 - ICMP attack detected! 1319 pings received from 192.168.5.130 within 1 second.
2025-06-05 19:19:05,435 - ICMP attack detected! 1317 pings received from 192.168.5.128 within 1 second.
```

5.3 ICMP Flooding Attack Test

In this test, a basic ICMP flooding attack was simulated using the following command from a Kali Linux virtual machine:

```
sudo ping -f 192.168.5.130
```

The attack rapidly sent hundreds of ICMP Echo Request packets to the target. SemiGuard successfully detected this abnormal traffic pattern by measuring the number of ICMP packets received within one second. As shown in **Figure 8**, more than 1600 pings from **IP 192.168.5.128** were detected in under one second.

In response, the system:

I. Logged the event in icmp_attack_log.txt
II. Triggered and sent an alert email to the administrator.
III. Displayed a real-time warning in the GUI panel

This test validated SemiGuard's ability to detect high-volume ICMP flooding attempts and notify the user promptly.

5.4 TCP SYN Flood (DDoS) Attack Test

To simulate a SYN flood attack, tools such as Nmap and Hping3 were used on a Kali Linux virtual machine. The commands below initiated high volumes of half-open TCP connections to the target (192.168.5.130):

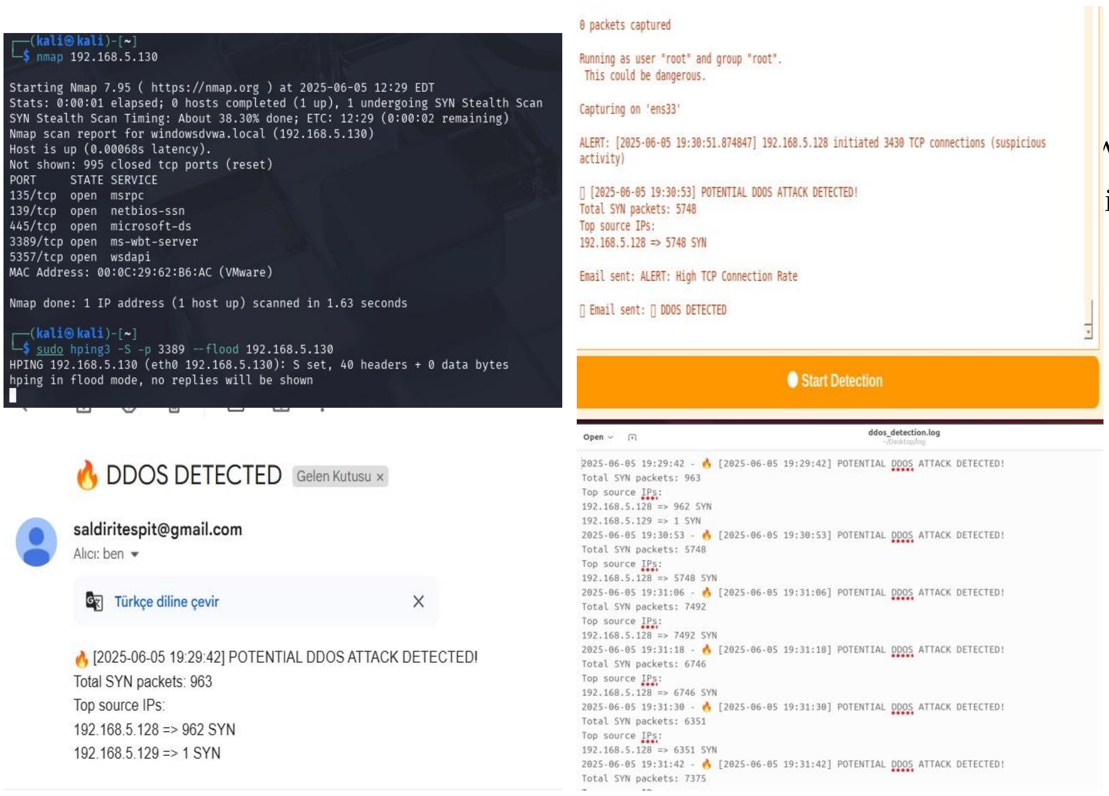
```
nmap -sS 192.168.5.130
sudo hping3 -S -p 3389 --flood
192.168.5.130
```

The SemiGuard system monitored incoming TCP SYN packets and identified a potential DDoS attack when the packet count exceeded a threshold. As shown in **Figure 9**, more than 7000 SYN packets from 192.168.5.128 were recorded within a short time interval.

The system responded with the following actions:

I. Logged the incident in ddos_detection.log
II. Sent an automated alert email including the top offending IPs
III. Displayed the detection message in the GUI in real time

This test proved SemiGuard can detect SYN flood attacks and instantly alert administrators.



```
┌──(kali㉿kali)-[~]
└─$ nmap 192.168.5.130

Starting Nmap 7.95 ( https://nmap.org ) at 2025-06-05 12:29 EDT
Stats: 0:00:01 elapsed; 0 hosts completed (1 up), 1 undergoing SYN Stealth Scan
SYN Stealth Scan Timing: About 38.30% done; ETC: 12:29 (0:00:02 remaining)
Nmap scan report for windowsdvwa.local (192.168.5.130)
Host is up (0.00068s latency).
Not shown: 995 closed tcp ports (reset)
PORT     STATE SERVICE
135/tcp  open  msrpc
139/tcp  open  netbios-ssn
445/tcp  open  microsoft-ds
3389/tcp open  ms-wbt-server
5357/tcp open  wsdapi
MAC Address: 00:0C:29:62:B6:AC (VMware)

Nmap done: 1 IP address (1 host up) scanned in 1.63 seconds

┌──(kali㉿kali)-[~]
└─$ sudo hping3 -S -p 3389 --flood 192.168.5.130
HPING 192.168.5.130 (eth0 192.168.5.130): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

```
0 packets captured

Running as user "root" and group "root".
 This could be dangerous.

Capturing on 'ens33'

ALERT: [2025-06-05 19:30:51.874847] 192.168.5.128 initiated 3430 TCP connections (suspicious
activity)

[] [2025-06-05 19:30:53] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 5748
Top source IPs:
192.168.5.128 => 5748 SYN

Email sent: ALERT: High TCP Connection Rate

[] Email sent: [] DDOS DETECTED
```

🔥 DDOS DETECTED Gelen Kutusu ×

saldiritespit@gmail.com
Alıcı: ben

🔥 [2025-06-05 19:29:42] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 963
Top source IPs:
192.168.5.128 => 962 SYN
192.168.5.129 => 1 SYN

```
2025-06-05 19:29:42 - 🔥 [2025-06-05 19:29:42] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 963
Top source IPs:
192.168.5.128 => 962 SYN
192.168.5.129 => 1 SYN
2025-06-05 19:30:53 - 🔥 [2025-06-05 19:30:53] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 5748
Top source IPs:
192.168.5.128 => 5748 SYN
2025-06-05 19:31:06 - 🔥 [2025-06-05 19:31:06] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 7492
Top source IPs:
192.168.5.128 => 7492 SYN
2025-06-05 19:31:18 - 🔥 [2025-06-05 19:31:18] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 6746
Top source IPs:
192.168.5.128 => 6746 SYN
2025-06-05 19:31:30 - 🔥 [2025-06-05 19:31:30] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 6351
Top source IPs:
192.168.5.128 => 6351 SYN
2025-06-05 19:31:42 - 🔥 [2025-06-05 19:31:42] POTENTIAL DDOS ATTACK DETECTED!
Total SYN packets: 7375
```

web

ing

Figure 9 – Execution and detection of TCP SYN Flood (DDoS) attack with log and email alert

5.5 Brute-force Attack Test

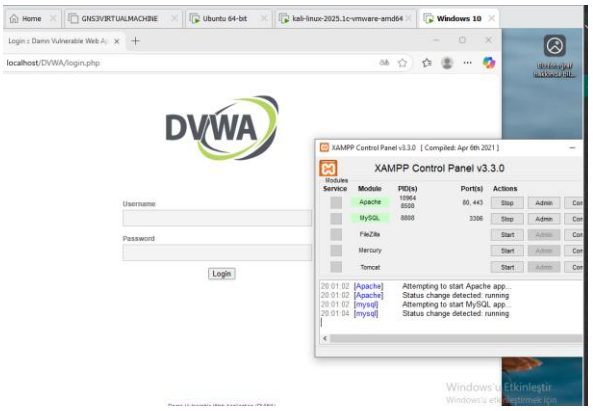
5.5.1 DVWA Environment Setup (Preliminary Preparation for Brute-force Testing)

To avoid launching brute-force attacks on real or unauthorized systems, a safe and controlled testing environment was established. For this purpose, the open-source “Damn Vulnerable Web Application” (DVWA) was deployed on a Windows-based virtual machine using the XAMPP server package, which includes Apache and MySQL services (**Figure 10**).

and educational use, and is publicly available at: https://github.com/digininja/DVWA

This setup enabled password guessing attacks to be performed against a login form in a local environment, ensuring that the experimental test process was conducted without violating any ethical or legal boundaries.

Figure 10 – DVWA interface and XAMPP configuration on the simulated victim machine



Module	PID(s)	Port(s)
Apache	10964 8508	80, 443
MySQL	8808	3306
FileZilla		
Mercury		
Tomcat		

```
20:01:02  [Apache]   Attempting to start Apache app...
20:01:02  [Apache]   Status change detected: running
20:01:02  [mysql]    Attempting to start MySQL app...
20:01:04  [mysql]    Status change detected: running
```

5.5.2 Brute-force Attack Execution and Detection

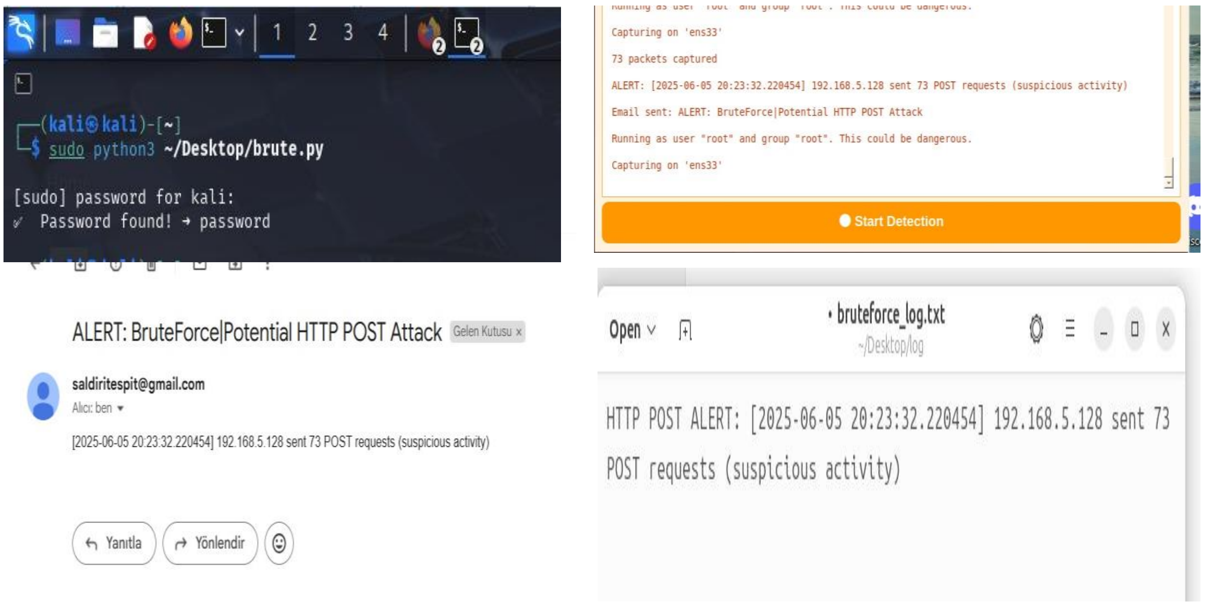
At this stage, a password guessing (brute-force) attack was conducted against the previously configured DVWA application. Assuming the username "admin" was known in advance, the attack was executed using a custom Python script named brute.py. This script reads each password from a given wordlist and submits it via a POST request to the login form. For each attempt, it dynamically retrieves the CSRF token from the login page to maintain a valid session.

As the attack was launched from a Kali Linux machine, 73 POST requests were sent to the target system within seconds.

This abnormal activity was successfully detected by the SemiGuard application. As shown in **Figure 11**:

I. The incident was logged in the brute_force_log.txt file
II. An alert email titled "BruteForce|Potential HTTP POST Attack" was sent to the system administrator
III. A real-time warning message was displayed in the GUI panel

The brute.py script identified the correct password once the response no longer contained the "Login failed" message, and the terminal output confirmed a successful login. This test demonstrated SemiGuard's capability to detect brute-force attacks in real time and respond with appropriate alerts and logging.



```
┌──(kali㉿kali)-[~]
└─$ sudo python3 ~/Desktop/brute.py

[sudo] password for kali:
✓  Password found! → password
```

```
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ens33'
73 packets captured
ALERT: [2025-06-05 20:23:32.220454] 192.168.5.128 sent 73 POST requests (suspicious activity)
Email sent: ALERT: BruteForce|Potential HTTP POST Attack
Running as user "root" and group "root". This could be dangerous.
Capturing on 'ens33'
```

ALERT: BruteForce|Potential HTTP POST Attack

saldiritespit@gmail.com
Alıcı: ben

[2025-06-05 20:23:32.220454] 192.168.5.128 sent 73 POST requests (suspicious activity)

```
HTTP POST ALERT: [2025-06-05 20:23:32.220454] 192.168.5.128 sent 73
POST requests (suspicious activity)
```

Figure 11 – Brute-force attack detection: terminal, GUI, email, and log output

6. CONCLUSION

The desktop-based intrusion detection system developed under this project, named SemiGuard, has successfully identified four types of cyber attacks—namely ARP Spoofing, ICMP Flooding, TCP SYN Flood, and Brute-force—in a virtual network environment. The application features a user-friendly graphical interface, logs each detected attack into separate files, and sends real-time email notifications to the system administrator. The test scenarios conducted throughout the project confirm that SemiGuard provides a reliable detection mechanism for fundamental attack types and can be considered a suitable tool for educational and experimental use cases.

However, the system also presents certain limitations. Most notably, it is designed to detect only four specific types of attacks using static Python scripts. As a result, it may fall short in identifying modern and more sophisticated threats, which have grown increasingly complex in recent years. For example, attacks such as advanced persistent threats (APT), DNS tunneling, SQL injection, or cross-site scripting (XSS) are beyond the current detection capabilities of the system. Moreover, since detection thresholds are predefined and fixed, performance may vary across different network environments, reducing its overall adaptability.

Future work should focus on increasing the flexibility and intelligence of the system by integrating machine learning-based anomaly detection and expanding the scope of detectable attack types. Additionally, implementing automated preventive measures, such as network isolation or access denial, would enable the system to adopt a more proactive security posture.

In conclusion, SemiGuard provides a functional baseline for detecting basic network attacks. Nonetheless, addressing the evolving landscape of cyber threats will require more adaptive, intelligent, and extensible IDS architectures.

REFERENCES

[1] Kryvyi Rih SPU & Borys Grinchenko KMU, *Simulation of ARP Attacks in GNS3 Using XArp*, CEUR Workshop Proceedings, 2023.

[2] N. Singh, S. Jain, *Using GNS3 for Simulating DDoS Attacks on HTTP Services*, ResearchGate, 2018.

[3] R. Basnet, *TCP SYN Flood Detection Based on Payload Analysis*, Journal of Computer Research, 2009.

[4] M. Shankaraiah et al., *Dynamic Network Intrusion Detection System for Virtual Machine Environment*, ResearchGate, 2023.

[5] Bordi Network Security Lab, *DoS Attacks – Network Security Course Project*, GitHub Repository, 2023.