SOC Analyst Project: CHECKER

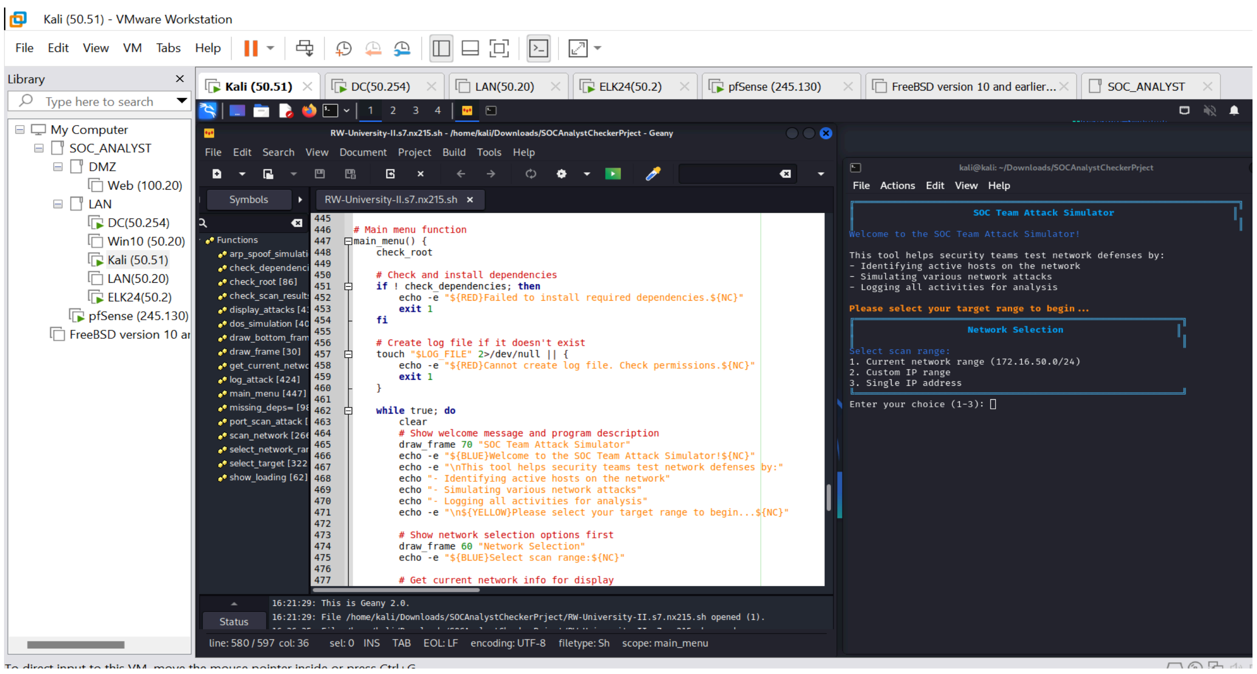
Program Code: NX220

Student Information

- **Name:** EMMANUEL SHYIRAMBERE
- **Student Code:** s7
- **Unit:** RW-University-II
- **Lecturer:** Mr. DOMINIC HARELIMANA

Project Overview

The SOC Analyst Checker project introduces a sophisticated security testing system designed to enhance the capabilities of Security Operations Center (SOC) teams. At its core, this tool enables security professionals to conduct controlled attack simulations while maintaining comprehensive logs of all activities, thereby improving team readiness and vulnerability assessment capabilities.


Kali (50.51) - VMware Workstation
Library
My Computer
SOC_ANALYST
DMZ
Web (100.20)
LAN
DC(50.254)
Win10 (50.20)
Kali (50.51)
LAN(50.20)
ELK24(50.2)
pfSense (245.130)
FreeBSD version 10 and earlier...
RW-University-II.s7.nx215.sh - /home/kali/Downloads/SOCAnalystCheckerPrject - Geany
kali@kali: ~/Downloads/SOCAnalystCheckerPrject

```
# Main menu function
main_menu() {
    check_root

    # Check and install dependencies
    if ! check_dependencies; then
        echo -e "${RED}Failed to install required dependencies.${NC}"
        exit 1
    fi

    # Create log file if it doesn't exist
    touch "$LOG_FILE" 2>/dev/null || {
        echo -e "${RED}Cannot create log file. Check permissions.${NC}"
        exit 1
    }

    while true; do
        clear
        # Show welcome message and program description
        draw_frame 70 "SOC Team Attack Simulator"
        echo -e "${BLUE}Welcome to the SOC Team Attack Simulator!${NC}"
        echo -e "\nThis tool helps security teams test network defenses by:"
        echo "- Identifying active hosts on the network"
        echo "- Simulating various network attacks"
        echo "- Logging all activities for analysis"
        echo -e "\n${YELLOW}Please select your target range to begin...${NC}"

        # Show network selection options first
        draw_frame 60 "Network Selection"
        echo -e "${BLUE}Select scan range:${NC}"

        # Get current network info for display
```

```
16:21:29: This is Geany 2.0.
16:21:29: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh opened (1).
line: 580 / 597  col: 36   sel: 0  INS  TAB  EOL: LF  encoding: UTF-8  filetype: Sh  scope: main_menu
```

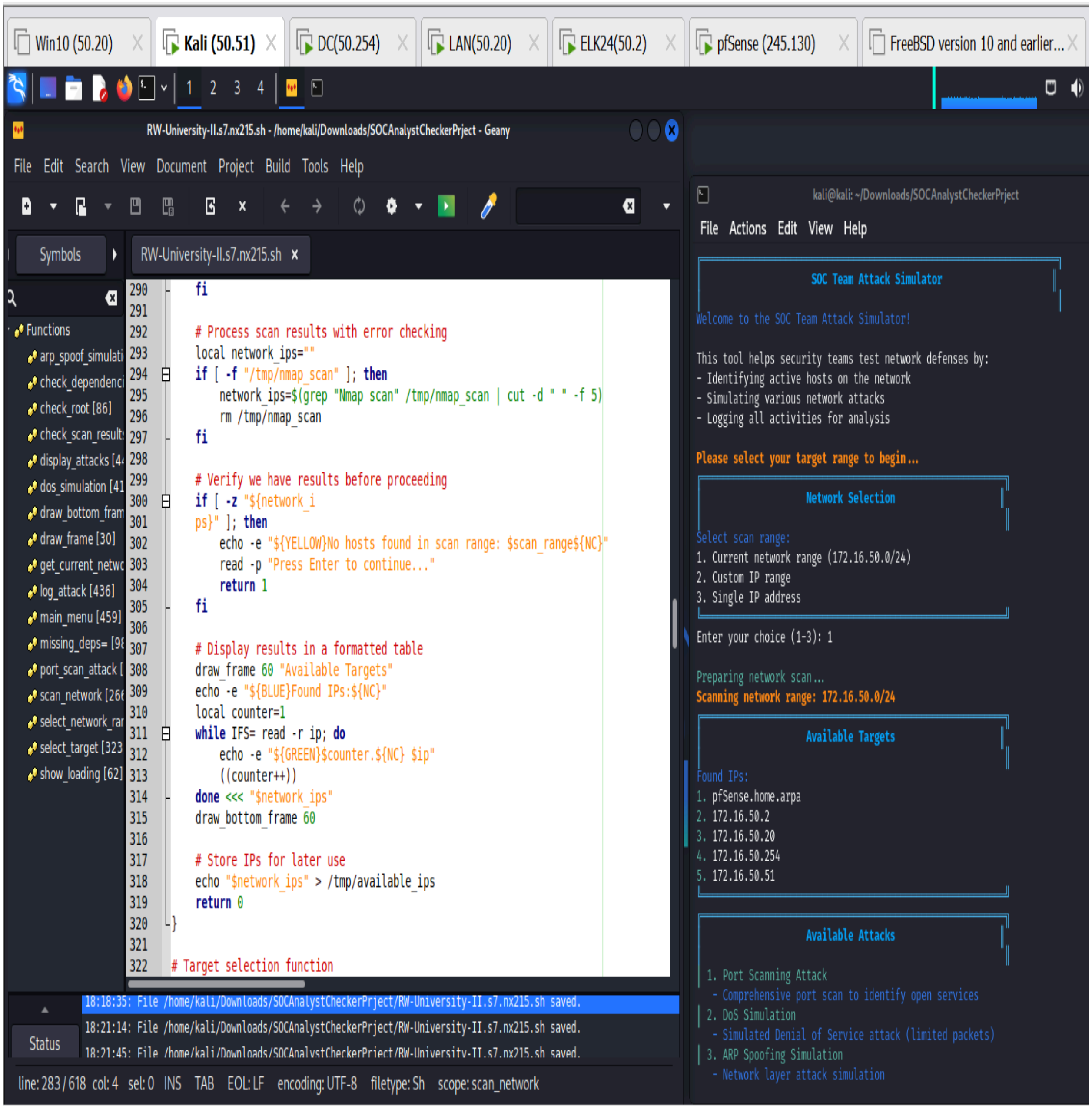
```
                    SOC Team Attack Simulator
Welcome to the SOC Team Attack Simulator!

This tool helps security teams test network defenses by:
- Identifying active hosts on the network
- Simulating various network attacks
- Logging all activities for analysis

Please select your target range to begin...
                 Network Selection
Select scan range:
1. Current network range (172.16.50.0/24)
2. Custom IP range
3. Single IP address

Enter your choice (1-3):
```

Screenshot: Initial system interface showing the main menu

System Architecture and Implementation

Network Discovery and Target Selection

The system begins its operation by performing a comprehensive network scan to identify available targets. This automated discovery process utilizes Nmap to safely enumerate network hosts, presenting the results in a clear, organized format. Users can either select specific targets or opt for random selection, providing flexibility in testing scenarios.



```
    fi

    # Process scan results with error checking
    local network_ips=""
    if [ -f "/tmp/nmap_scan" ]; then
        network_ips=$(grep "Nmap scan" /tmp/nmap_scan | cut -d " " -f 5)
        rm /tmp/nmap_scan
    fi

    # Verify we have results before proceeding
    if [ -z "${network_i
ps}" ]; then
        echo -e "${YELLOW}No hosts found in scan range: $scan_range${NC}"
        read -p "Press Enter to continue..."
        return 1
    fi

    # Display results in a formatted table
    draw_frame 60 "Available Targets"
    echo -e "${BLUE}Found IPs:${NC}"
    local counter=1
    while IFS= read -r ip; do
        echo -e "${GREEN}$counter.${NC} $ip"
        ((counter++))
    done <<< "$network_ips"
    draw_bottom_frame 60

    # Store IPs for later use
    echo "$network_ips" > /tmp/available_ips
    return 0
}

# Target selection function
```

```
18:18:35: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:14: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:45: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
```

```
line: 283 / 618   col: 4   sel: 0   INS   TAB   EOL: LF   encoding: UTF-8   filetype: Sh   scope: scan_network
```

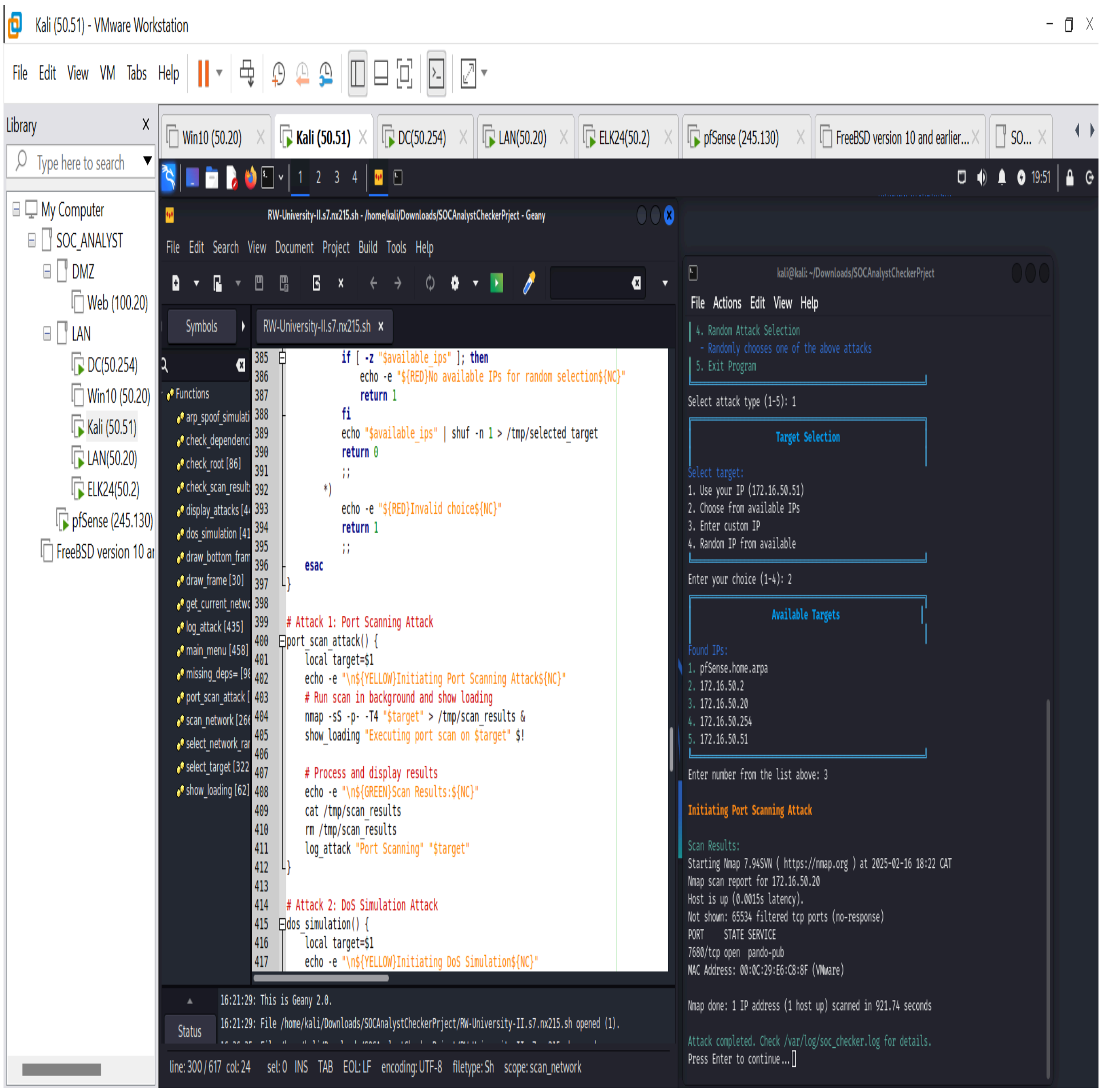
```
              SOC Team Attack Simulator

Welcome to the SOC Team Attack Simulator!

This tool helps security teams test network defenses by:
- Identifying active hosts on the network
- Simulating various network attacks
- Logging all activities for analysis

Please select your target range to begin...

              Network Selection

Select scan range:
1. Current network range (172.16.50.0/24)
2. Custom IP range
3. Single IP address

Enter your choice (1-3): 1

Preparing network scan...
Scanning network range: 172.16.50.0/24

              Available Targets

Found IPs:
1. pfSense.home.arpa
2. 172.16.50.2
3. 172.16.50.20
4. 172.16.50.254
5. 172.16.50.51

              Available Attacks

 1. Port Scanning Attack
  - Comprehensive port scan to identify open services
 2. DoS Simulation
  - Simulated Denial of Service attack (limited packets)
 3. ARP Spoofing Simulation
  - Network layer attack simulation
```

Screenshot: Network scanning process showing the discovery of available hosts

Attack Simulation Capabilities

The system implements three distinct types of security tests, each designed to evaluate different aspects of network security:

Port Scanning Mechanism This primary testing capability employs Nmap to conduct thorough port analysis of target systems. The scan provides detailed information about open services and potential vulnerabilities, essential for security assessment. The implementation includes progress monitoring and result formatting for clarity.



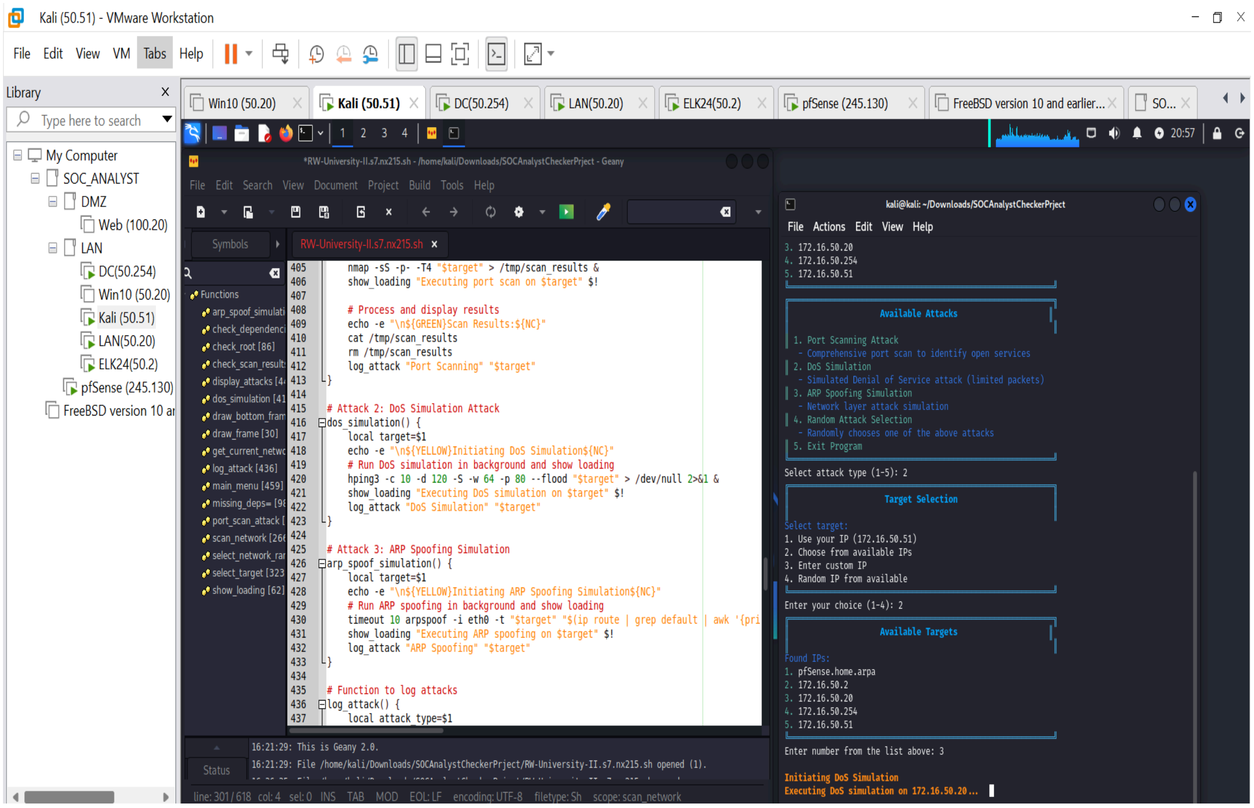
```
            if [ -z "$available_ips" ]; then
                echo -e "${RED}No available IPs for random selection${NC}"
                return 1
            fi
            echo "$available_ips" | shuf -n 1 > /tmp/selected_target
            return 0
            ;;
        *)
            echo -e "${RED}Invalid choice${NC}"
            return 1
            ;;
    esac
}

# Attack 1: Port Scanning Attack
port_scan_attack() {
    local target=$1
    echo -e "\n${YELLOW}Initiating Port Scanning Attack${NC}"
    # Run scan in background and show loading
    nmap -sS -p- -T4 "$target" > /tmp/scan_results &
    show_loading "Executing port scan on $target" $!

    # Process and display results
    echo -e "\n${GREEN}Scan Results:${NC}"
    cat /tmp/scan_results
    rm /tmp/scan_results
    log_attack "Port Scanning" "$target"
}

# Attack 2: DoS Simulation Attack
dos_simulation() {
    local target=$1
    echo -e "\n${YELLOW}Initiating DoS Simulation${NC}"
```

```
16:21:29: This is Geany 2.0.
16:21:29: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh opened (1).
```

```
| 4. Random Attack Selection
   - Randomly chooses one of the above attacks
| 5. Exit Program

Select attack type (1-5): 1

                Target Selection

Select target:
1. Use your IP (172.16.50.51)
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

Enter your choice (1-4): 2

               Available Targets

Found IPs:
1. pfSense.home.arpa
2. 172.16.50.2
3. 172.16.50.20
4. 172.16.50.254
5. 172.16.50.51

Enter number from the list above: 3

Initiating Port Scanning Attack

Scan Results:
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-02-16 18:22 CAT
Nmap scan report for 172.16.50.20
Host is up (0.0015s latency).
Not shown: 65534 filtered tcp ports (no-response)
PORT     STATE SERVICE
7680/tcp open  pando-pub
MAC Address: 00:0C:29:E6:C8:8F (VMware)

Nmap done: 1 IP address (1 host up) scanned in 921.74 seconds

Attack completed. Check /var/log/soc_checker.log for details.
Press Enter to continue...
```

Screenshot: Port scanning execution showing progress and results

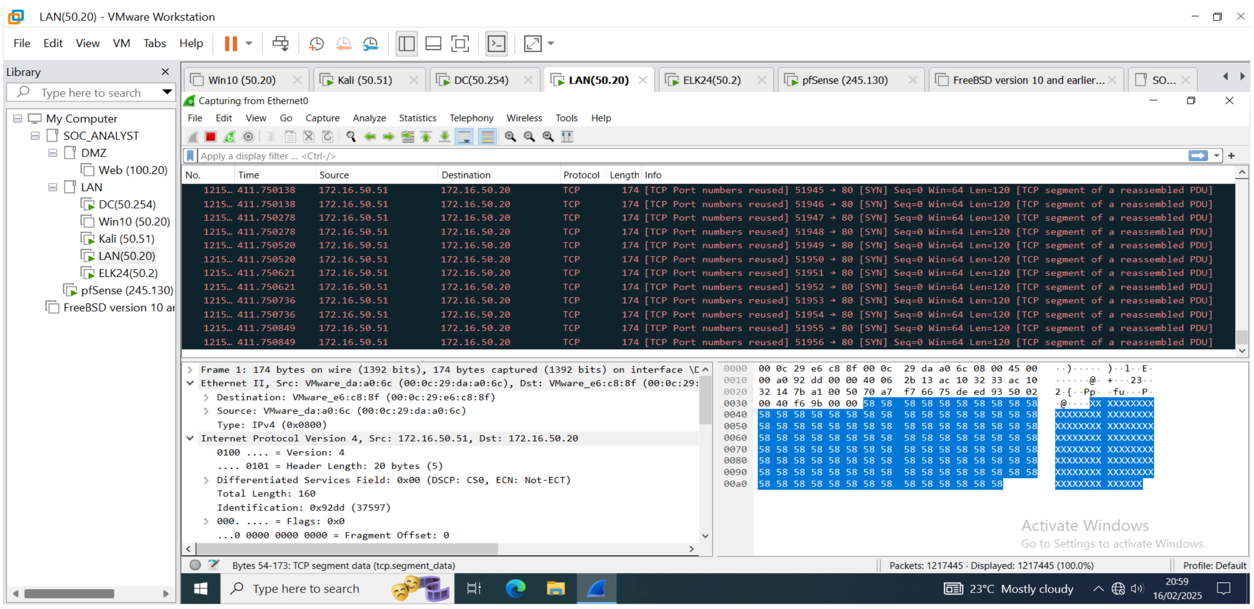
Denial of Service Testing The DoS simulation feature utilizes hping3 to conduct controlled testing of system resilience. This implementation carefully limits packet transmission to prevent actual service disruption while still providing valuable insights into system response characteristics.



```
        nmap -sS -p- -T4 "$target" > /tmp/scan_results &
        show_loading "Executing port scan on $target" $!

        # Process and display results
        echo -e "\n${GREEN}Scan Results:${NC}"
        cat /tmp/scan_results
        rm /tmp/scan_results
        log_attack "Port Scanning" "$target"
}

# Attack 2: DoS Simulation Attack
dos_simulation() {
        local target=$1
        echo -e "\n${YELLOW}Initiating DoS Simulation${NC}"
        # Run DoS simulation in background and show loading
        hping3 -c 10 -d 120 -S -w 64 -p 80 --flood "$target" > /dev/null 2>&1 &
        show_loading "Executing DoS simulation on $target" $!
        log_attack "DoS Simulation" "$target"
}

# Attack 3: ARP Spoofing Simulation
arp_spoof_simulation() {
        local target=$1
        echo -e "\n${YELLOW}Initiating ARP Spoofing Simulation${NC}"
        # Run ARP spoofing in background and show loading
        timeout 10 arpspoof -i eth0 -t "$target" "$(ip route | grep default | awk '{pri
        show_loading "Executing ARP spoofing on $target" $!
        log_attack "ARP Spoofing" "$target"
}

# Function to log attacks
log_attack() {
        local attack_type=$1
```

```
16:21:29: This is Geany 2.0.
16:21:29: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh opened (1).
```

```
3. 172.16.50.20
4. 172.16.50.254
5. 172.16.50.51

                 Available Attacks

 1. Port Scanning Attack
   - Comprehensive port scan to identify open services
 2. DoS Simulation
   - Simulated Denial of Service attack (limited packets)
 3. ARP Spoofing Simulation
   - Network layer attack simulation
 4. Random Attack Selection
   - Randomly chooses one of the above attacks
 5. Exit Program

Select attack type (1-5): 2

                 Target Selection

Select target:
1. Use your IP (172.16.50.51)
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

Enter your choice (1-4): 2

                 Available Targets

Found IPs:
1. pfSense.home.arpa
2. 172.16.50.2
3. 172.16.50.20
4. 172.16.50.254
5. 172.16.50.51

Enter number from the list above: 3

Initiating DoS Simulation
Executing DoS simulation on 172.16.50.20...
```



```
No.    Time        Source        Destination   Protocol Length Info
1215… 411.750138  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51945 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750138  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51946 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750278  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51947 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750278  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51948 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750520  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51949 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750520  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51950 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750621  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51951 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750621  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51952 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750736  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51953 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750736  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51954 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750849  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51955 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
1215… 411.750849  172.16.50.51  172.16.50.20  TCP      174 [TCP Port numbers reused] 51956 → 80 [SYN] Seq=0 Win=64 Len=120 [TCP segment of a reassembled PDU]
```

```
> Frame 1: 174 bytes on wire (1392 bits), 174 bytes captured (1392 bits) on interface \D
v Ethernet II, Src: VMware_da:a0:6c (00:0c:29:da:a0:6c), Dst: VMware_e6:c8:8f (00:0c:29:
  > Destination: VMware_e6:c8:8f (00:0c:29:e6:c8:8f)
  > Source: VMware_da:a0:6c (00:0c:29:da:a0:6c)
    Type: IPv4 (0x0800)
v Internet Protocol Version 4, Src: 172.16.50.51, Dst: 172.16.50.20
    0100 .... = Version: 4
    .... 0101 = Header Length: 20 bytes (5)
  > Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
    Total Length: 160
    Identification: 0x92dd (37597)
  > 000. .... = Flags: 0x0
    ...0 0000 0000 0000 = Fragment Offset: 0
```

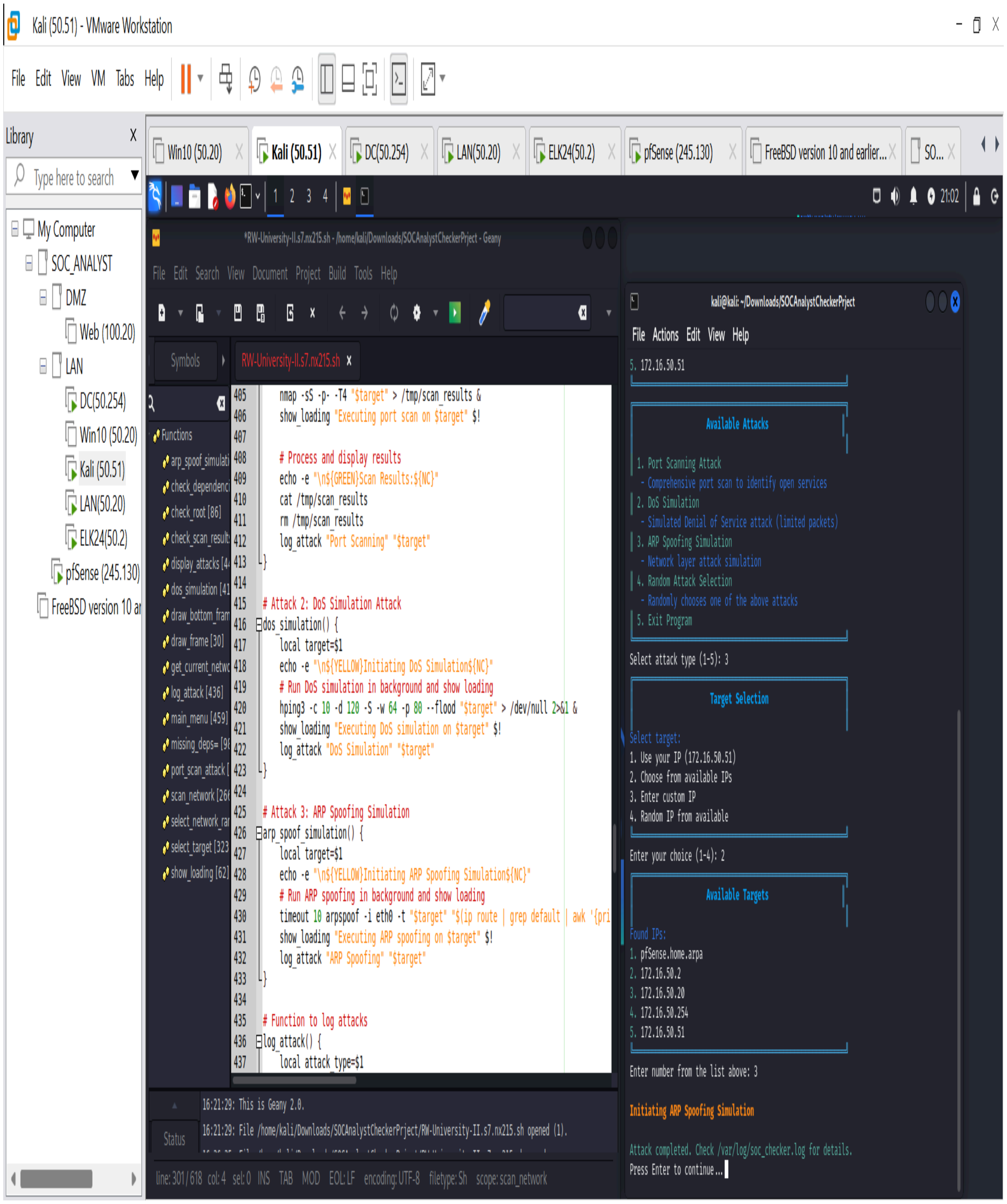
```
0000  00 0c 29 e6 c8 8f 00 0c  29 da a0 6c 08 00 45 00   ··)·····  )··l··E·
0010  00 a0 92 dd 00 00 40 06  2b 13 ac 10 32 33 ac 10   ······@·  +···23··
0020  32 14 7b a1 00 50 70 a7  f7 66 75 de ed 93 50 02   2·{··Pp·  ·fu···P·
0030  00 40 f6 9b 00 00 58 58  58 58 58 58 58 58 58 58   ·@····XX  XXXXXXXX
0040  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
0050  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
0060  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
0070  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
0080  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
0090  58 58 58 58 58 58 58 58  58 58 58 58 58 58 58 58   XXXXXXXX  XXXXXXXX
00a0  58 58 58 58 58 58 58 58  58 58 58 58 58 58         XXXXXXXX  XXXXXX
```

Bytes 54-173: TCP segment data (tcp.segment_data)

Packets: 1217445 · Displayed: 1217445 (100.0%)

Screenshot: DoS simulation interface showing controlled execution

ARP Spoofing Assessment Network layer security testing is accomplished through controlled ARP spoofing simulations. This feature helps validate network monitoring capabilities and tests the effectiveness of ARP poisoning detection systems.


Kali (50.51) - VMware Workstation
File Edit View VM Tabs Help
Library
Type here to search
My Computer
SOC_ANALYST
DMZ
Web (100.20)
LAN
DC(50.254)
Win10 (50.20)
Kali (50.51)
LAN(50.20)
ELK24(50.2)
pfSense (245.130)
FreeBSD version 10 ar
*RW-University-II.s7.nx215.sh - /home/kali/Downloads/SOCAnalystCheckerPrject - Geany
RW-University-II.s7.nx215.sh
Functions
kali@kali: ~/Downloads/SOCAnalystCheckerPrject

```
    nmap -sS -p- -T4 "$target" > /tmp/scan_results &
    show_loading "Executing port scan on $target" $!

    # Process and display results
    echo -e "\n${GREEN}Scan Results:${NC}"
    cat /tmp/scan_results
    rm /tmp/scan_results
    log_attack "Port Scanning" "$target"
}

# Attack 2: DoS Simulation Attack
dos_simulation() {
    local target=$1
    echo -e "\n${YELLOW}Initiating DoS Simulation${NC}"
    # Run DoS simulation in background and show loading
    hping3 -c 10 -d 120 -S -w 64 -p 80 --flood "$target" > /dev/null 2>&1 &
    show_loading "Executing DoS simulation on $target" $!
    log_attack "DoS Simulation" "$target"
}

# Attack 3: ARP Spoofing Simulation
arp_spoof_simulation() {
    local target=$1
    echo -e "\n${YELLOW}Initiating ARP Spoofing Simulation${NC}"
    # Run ARP spoofing in background and show loading
    timeout 10 arpspoof -i eth0 -t "$target" "$(ip route | grep default | awk '{pri
    show_loading "Executing ARP spoofing on $target" $!
    log_attack "ARP Spoofing" "$target"
}

# Function to log attacks
log_attack() {
    local attack_type=$1
```

```
16:21:29: This is Geany 2.0.
16:21:29: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh opened (1).
```

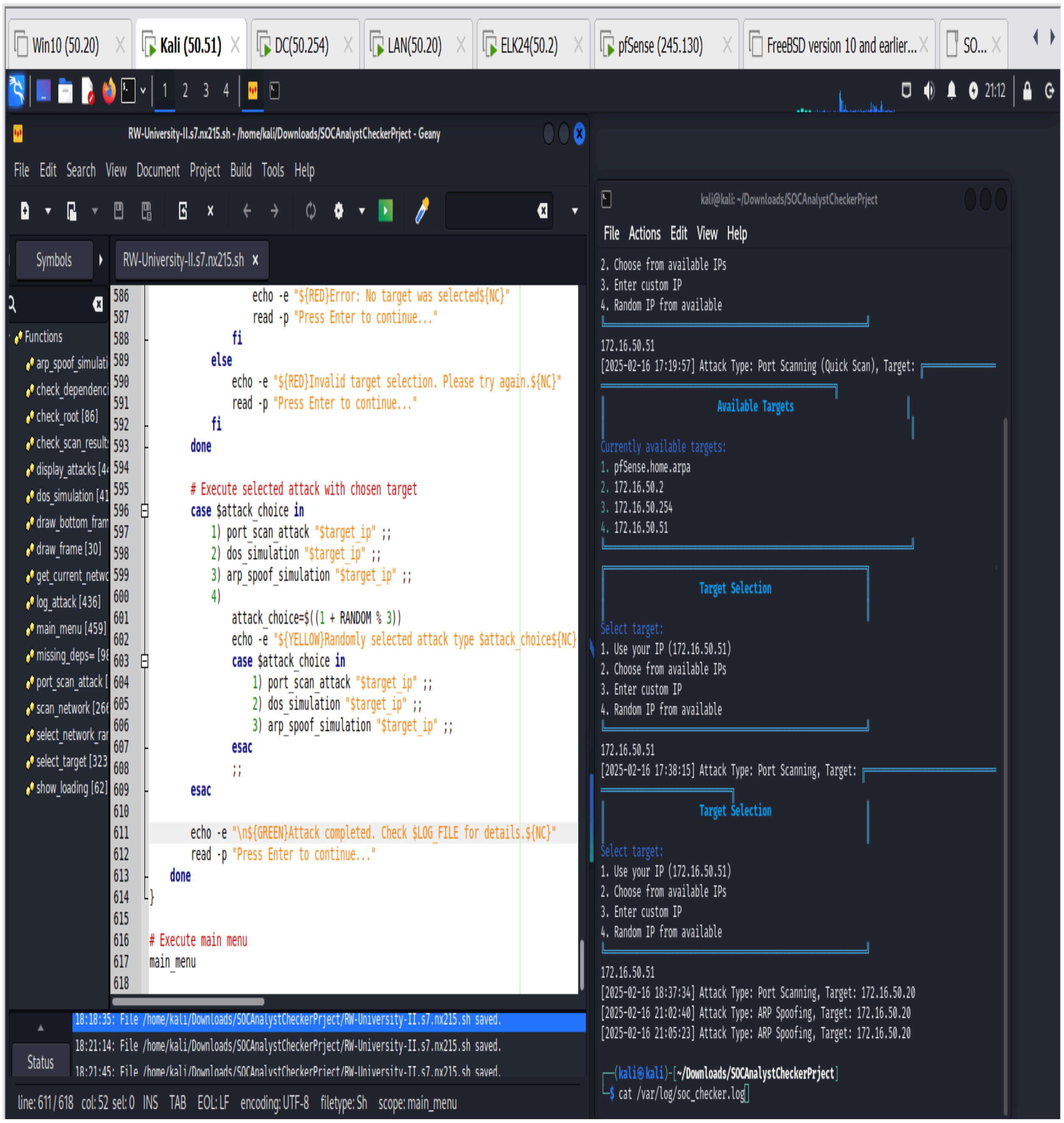
```
line: 301 / 618  col: 4  sel: 0  INS  TAB  MOD  EOL: LF  encoding: UTF-8  filetype: Sh  scope: scan_network
```

```
5. 172.16.50.51

              Available Attacks

 1. Port Scanning Attack
  - Comprehensive port scan to identify open services
 2. DoS Simulation
  - Simulated Denial of Service attack (limited packets)
 3. ARP Spoofing Simulation
  - Network layer attack simulation
 4. Random Attack Selection
  - Randomly chooses one of the above attacks
 5. Exit Program

Select attack type (1-5): 3

              Target Selection

Select target:
1. Use your IP (172.16.50.51)
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

Enter your choice (1-4): 2

              Available Targets

Found IPs:
1. pfSense.home.arpa
2. 172.16.50.2
3. 172.16.50.20
4. 172.16.50.254
5. 172.16.50.51

Enter number from the list above: 3

Initiating ARP Spoofing Simulation

Attack completed. Check /var/log/soc_checker.log for details.
Press Enter to continue...
```

Screenshot: ARP spoofing test execution with real-time status updates

Logging and Monitoring

Every operation within the system is meticulously logged, creating a detailed audit trail of all testing activities. The logging system captures timestamp information, attack types, and target details, storing them in a standardized format for easy analysis.



```
                echo -e "${RED}Error: No target was selected${NC}"
                read -p "Press Enter to continue..."
            fi
        else
            echo -e "${RED}Invalid target selection. Please try again.${NC}"
            read -p "Press Enter to continue..."
        fi
    done

    # Execute selected attack with chosen target
    case $attack_choice in
        1) port_scan_attack "$target_ip" ;;
        2) dos_simulation "$target_ip" ;;
        3) arp_spoof_simulation "$target_ip" ;;
        4)
            attack_choice=$((1 + RANDOM % 3))
            echo -e "${YELLOW}Randomly selected attack type $attack_choice${NC}
            case $attack_choice in
                1) port_scan_attack "$target_ip" ;;
                2) dos_simulation "$target_ip" ;;
                3) arp_spoof_simulation "$target_ip" ;;
            esac
            ;;
    esac

    echo -e "\n${GREEN}Attack completed. Check $LOG_FILE for details.${NC}"
    read -p "Press Enter to continue..."
  done
}

# Execute main menu
main_menu
```

```
18:18:35: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:14: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:45: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
```

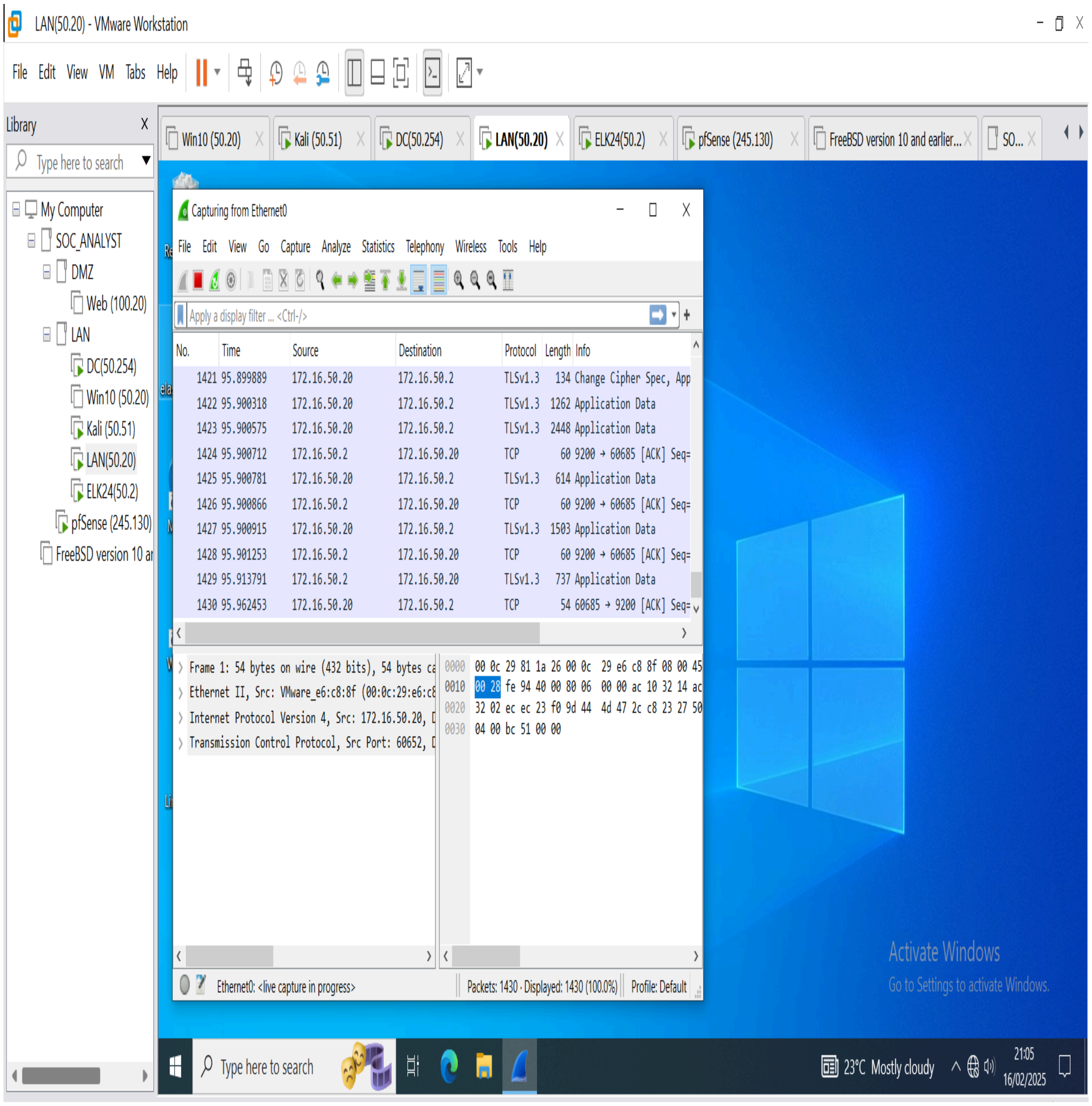
```
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

172.16.50.51
[2025-02-16 17:19:57] Attack Type: Port Scanning (Quick Scan), Target:

              Available Targets

Currently available targets:
1. pfSense.home.arpa
2. 172.16.50.2
3. 172.16.50.254
4. 172.16.50.51

              Target Selection

Select target:
1. Use your IP (172.16.50.51)
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

172.16.50.51
[2025-02-16 17:38:15] Attack Type: Port Scanning, Target:

              Target Selection

Select target:
1. Use your IP (172.16.50.51)
2. Choose from available IPs
3. Enter custom IP
4. Random IP from available

172.16.50.51
[2025-02-16 18:37:34] Attack Type: Port Scanning, Target: 172.16.50.20
[2025-02-16 21:02:40] Attack Type: ARP Spoofing, Target: 172.16.50.20
[2025-02-16 21:05:23] Attack Type: ARP Spoofing, Target: 172.16.50.20

┌──(kali㉿kali)-[~/Downloads/SOCAnalystCheckerPrject]
└─$ cat /var/log/soc_checker.log
```

Screenshot: Log file contents showing formatted entries of various attack executions

Testing and Validation

Environment Configuration

Testing was conducted in a controlled VMware Workstation environment, utilizing multiple network segments including a LAN Network (50.20). This setup provided a realistic testing ground while ensuring no impact on production systems.



```
No.   Time       Source        Destination   Protocol Length Info
 1421 95.899889  172.16.50.20  172.16.50.2   TLSv1.3   134 Change Cipher Spec, App
 1422 95.900318  172.16.50.20  172.16.50.2   TLSv1.3  1262 Application Data
 1423 95.900575  172.16.50.20  172.16.50.2   TLSv1.3  2448 Application Data
 1424 95.900712  172.16.50.2   172.16.50.20  TCP        60 9200 → 60685 [ACK] Seq=
 1425 95.900781  172.16.50.20  172.16.50.2   TLSv1.3   614 Application Data
 1426 95.900866  172.16.50.2   172.16.50.20  TCP        60 9200 → 60685 [ACK] Seq=
 1427 95.900915  172.16.50.20  172.16.50.2   TLSv1.3  1503 Application Data
 1428 95.901253  172.16.50.2   172.16.50.20  TCP        60 9200 → 60685 [ACK] Seq=
 1429 95.913791  172.16.50.2   172.16.50.20  TLSv1.3   737 Application Data
 1430 95.962453  172.16.50.20  172.16.50.2   TCP        54 60685 → 9200 [ACK] Seq=
```

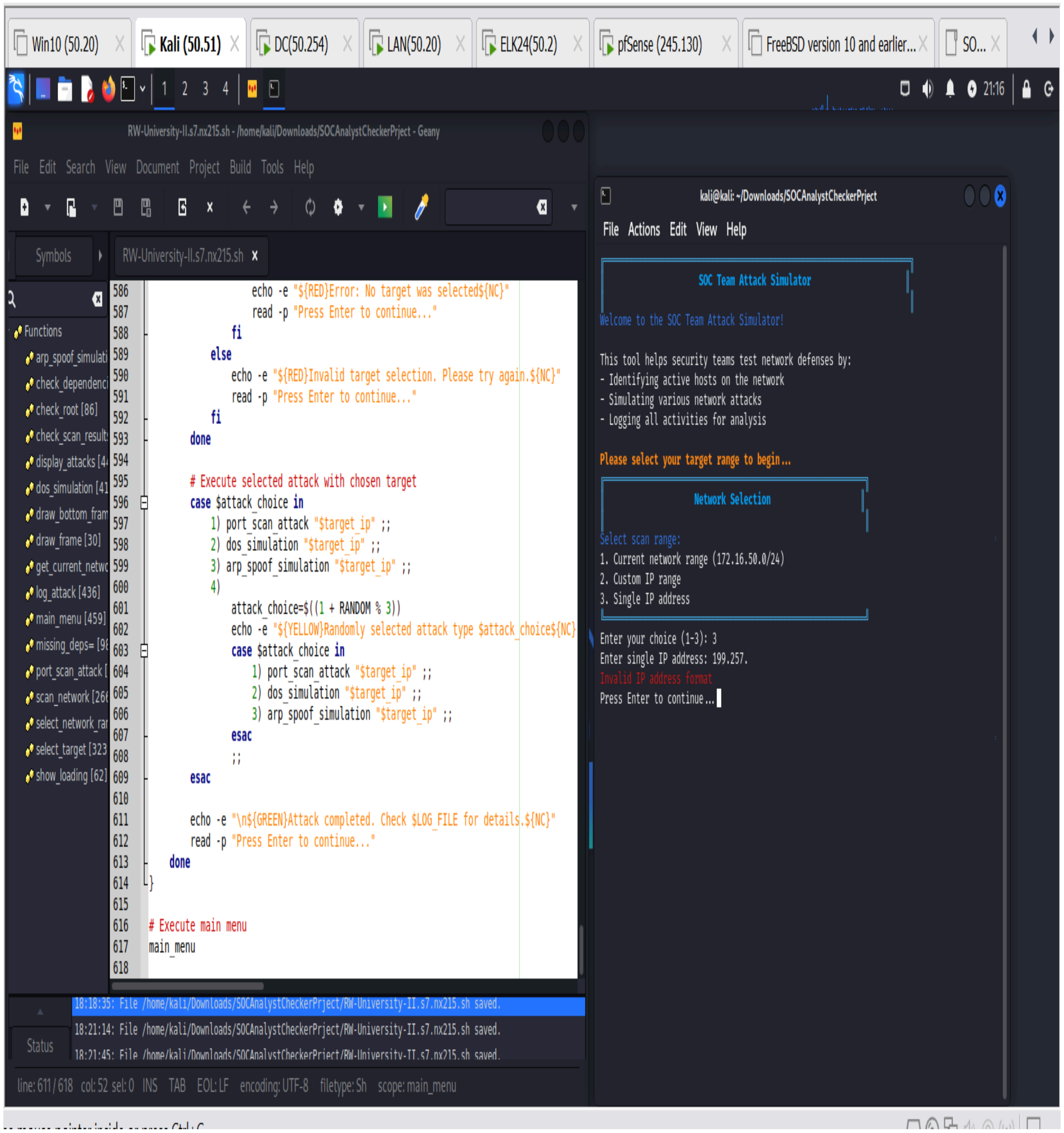
```
> Frame 1: 54 bytes on wire (432 bits), 54 bytes ca
> Ethernet II, Src: VMware_e6:c8:8f (00:0c:29:e6:c8
> Internet Protocol Version 4, Src: 172.16.50.20, D
> Transmission Control Protocol, Src Port: 60652, D
```

```
0000  00 0c 29 81 1a 26 00 0c  29 e6 c8 8f 08 00 45
0010  00 28 fe 94 40 00 80 06  00 00 ac 10 32 14 ac
0020  32 02 ec ec 23 f0 9d 44  4d 47 2c c8 23 27 50
0030  04 00 bc 51 00 00
```

Screenshot: VMware environment showing network configuration

Error Management

The system shows robust error handling capabilities, appropriately managing scenarios such as invalid input, insufficient privileges, and network connectivity issues. Each error condition triggers appropriate user notifications and system responses, maintaining operational stability.



```
            echo -e "${RED}Error: No target was selected${NC}"
            read -p "Press Enter to continue..."
          fi
        else
          echo -e "${RED}Invalid target selection. Please try again.${NC}"
          read -p "Press Enter to continue..."
        fi
      done

      # Execute selected attack with chosen target
      case $attack_choice in
        1) port_scan_attack "$target_ip" ;;
        2) dos_simulation "$target_ip" ;;
        3) arp_spoof_simulation "$target_ip" ;;
        4)
          attack_choice=$((1 + RANDOM % 3))
          echo -e "${YELLOW}Randomly selected attack type $attack_choice${NC}"
          case $attack_choice in
            1) port_scan_attack "$target_ip" ;;
            2) dos_simulation "$target_ip" ;;
            3) arp_spoof_simulation "$target_ip" ;;
          esac
          ;;
      esac

      echo -e "\n${GREEN}Attack completed. Check $LOG_FILE for details.${NC}"
      read -p "Press Enter to continue..."
    done
}

# Execute main menu
main_menu
```

```
18:18:35: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:14: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
18:21:45: File /home/kali/Downloads/SOCAnalystCheckerPrject/RW-University-II.s7.nx215.sh saved.
```

```
line: 611 / 618   col: 52  sel: 0   INS   TAB   EOL: LF   encoding: UTF-8   filetype: Sh   scope: main_menu
```

```
                SOC Team Attack Simulator

Welcome to the SOC Team Attack Simulator!

This tool helps security teams test network defenses by:
- Identifying active hosts on the network
- Simulating various network attacks
- Logging all activities for analysis

Please select your target range to begin...

              Network Selection

Select scan range:
1. Current network range (172.16.50.0/24)
2. Custom IP range
3. Single IP address

Enter your choice (1-3): 3
Enter single IP address: 199.257.
Invalid IP address format
Press Enter to continue...
```

Screenshot: Error handling demonstration showing user notification

Security Considerations and Future Development

The implementation maintains strong security practices throughout its operation. Root privilege verification, input validation, and careful attack parameter control ensure safe operation. All temporary files are properly managed, and comprehensive logging provides full activity accountability.

Looking forward, the system's modular design allows for future enhancements such as additional attack simulations, enhanced reporting capabilities, and potential integration with SIEM systems. The foundation has been laid for expanding the tool's capabilities while maintaining its user-friendly nature.

Conclusion

The SOC Analyst Checker project successfully delivers a comprehensive security testing platform that balances functionality with usability. Through careful implementation of attack simulations, robust logging, and user-friendly interfaces, the system provides SOC teams with a valuable tool for maintaining operational readiness and security awareness.