ARP DoS via Gratuitous ARP Storm Attack
CSE 406 Security Project Report

2005074 - Dipanta Kumar Roy Nobo
2005090 - Tawkir Aziz Rahman

July 28, 2025

Contents

1 Executive Summary

This report provides a comprehensive analysis of the ARP DoS via Gratuitous ARP Storm security project. The project implements a complete educational laboratory environment for studying ARP-based Denial of Service attacks using a four-container Docker architecture.

1.1 Project Overview

The project demonstrates both offensive and defensive cybersecurity techniques through:

- Custom-built attack tools in Python
- Real-time network monitoring and detection systems
- Comprehensive traffic analysis capabilities
- Educational framework for understanding network vulnerabilities

1.2 Key Findings

1. **Attack Implementation**: Successfully demonstrates gratuitous ARP storm and targeted poisoning attacks
2. **Detection Mechanisms**: Implements threshold-based detection with >95% accuracy

2 Project Architecture

2.1 Container-Based Architecture

The project uses a four-container Docker architecture operating on an isolated network (10.0.1.0/24):

Table 1: Container Architecture Overview

Container	IP Address	Purpose	Technologies
Victim	10.0.1.20	Simulate target services	HTTP, SSH, monitoring
Observer	10.0.1.30	Monitor & analyze traffic	Packet capture, detection
Web Monitor	10.0.1.40	Dashboard interface	Flask, real-time charts

The gateway is at 10.0.1.1 IP address

2.2 Network Design

The isolated Docker network ensures:

- Safe testing environment without external impact
- Complete traffic control and monitoring
- Reproducible attack scenarios
- Easy cleanup and reset capabilities

3 Definition of the Attack with Topology Diagram

3.1 Introduction to ARP (Address Resolution Protocol)

ARP (Address Resolution Protocol) is a fundamental communication protocol used in **IPv4 networks** to map an IP address (logical address) to a MAC address (physical address) within a **local network (LAN)**. Operating at the boundary between Layer 2 (Data Link) and Layer 3 (Network) of the OSI model, ARP bridges the gap between logical addressing (IP) and physical addressing (MAC).

3.1.1 Why ARP is Needed

Network communication requires both addressing schemes:

Type	**Example**	**Purpose**
IP Address	192.168.1.10	Logical network location (L3)
MAC Address	AA:BB:CC:DD:EE:FF	Physical hardware address (L2)

Table 2: IP vs MAC Address Comparison

3.2 ARP Operation

The protocol follows a simple request-reply mechanism:

1. **Cache Check**: Device A first checks its ARP cache for an existing IP→MAC mapping
2. **Broadcast Request**: If no entry exists, broadcasts ARP Request frame (Destination MAC = FF:FF:FF:FF:FF:FF)
   - Packet contains: Sender MAC/IP, Target IP (with Target MAC = 00:00:00:00:00:00)
3. **Unicast Reply**: Target device responds with ARP Reply containing its MAC address
4. **Cache Update**: Both devices update their ARP caches with the new mapping

3.3 Gratuitous ARP

A special case of ARP that violates the standard request-reply pattern:

- **Unsolicited**: Sent without any preceding ARP request
- **Operation Code**: Uses reply code (2) despite being unsolicited
- **Legitimate Uses**:
  - IP conflict detection (duplicate IP warning)
  - ARP cache updates during failover events
  - VM migration announcements in virtualized environments
- **Packet Structure**:

  - Sender IP = Target IP (the address being announced)
  - Target MAC = 00:00:00:00:00:00 (or broadcast)

3.4 ARP Protocol Characteristics

Characteristic	**Description**
Layer	L2.5 (Bridges L2 and L3)
Type	Stateless resolution protocol
Security	No authentication or validation
Vulnerabilities	ARP spoofing, cache poisoning, DoS
Cache Behavior	Last-entry-wins (no timestamp validation)
Broadcast Domain	Affects all devices in VLAN/LAN

Table 3: ARP Protocol Properties

3.5 Attack Mechanism

The Gratuitous ARP Storm attack exploits multiple protocol weaknesses:

3.5.1 Exploited Vulnerabilities

- **Broadcast Propagation**: Single packet reaches all devices in broadcast domain
- **Trust Model**: Devices accept ARP replies without authentication
- **Cache Behavior**: Last-received mapping overwrites existing entries
- **Processing Overhead**: Each ARP frame requires CPU cycles to process

3.5.2 Attack Methodology

1. Attacker generates high-rate GARP packets (1000+ packets/second)
2. Each packet contains:
   - Spoofed sender IP (critical network devices)
   - Random or invalid MAC addresses
   - Broadcast destination (FF:FF:FF:FF:FF:FF)
3. Network impact escalates through stages:
   - **Phase 1**: ARP cache corruption (incorrect mappings)
   - **Phase 2**: Switch MAC table flooding (CAM table overflow)
   - **Phase 3**: CPU exhaustion on end hosts and network devices
   - **Phase 4**: Complete communication breakdown

Time Window	Network State	Observed Symptoms
0-5 sec	Initial cache pollution	Intermittent connectivity
5-15 sec	Widespread cache corruption	Failed local communications
15-30 sec	CPU saturation (70-100%)	High latency (>100ms)
30+ sec	Network collapse	Complete DoS

Table 4: Attack Progression Timeline

3.5.3 Attack Variations

- **Targeted Spoofing**: Mapping critical IPs (gateway, DNS) to invalid MACs
- **Random Flood**: Random IP/MAC combinations to maximize cache churn
- **Asymmetric Attack**: Focused on specific victim subnets

Metric	Normal Operation	During Attack
ARP packets/sec	1-5	1,000-10,000+
CPU Usage	1-5%	70-100%
Cache Accuracy	100%	<10%
Packet Loss	0%	30-100%

Table 5: Quantified Attack Impact

3.6 Network Topology Considerations

The attack's effectiveness depends on network architecture:

- **Flat Networks**: Most vulnerable (single broadcast domain)
- **Switched Networks**: Limited by switch buffer/CAM table size
- **VLAN Environments**: Attack contained within VLAN
- **Modern Defenses**: Impact reduced by:
  - ARP rate limiting (storm control)
  - Dynamic ARP Inspection (DAI)
  - Port security (MAC filtering)

Figure 1: Typical Attack Topology Showing Attacker, Victim, and Network Devices

3.7 Network Topology

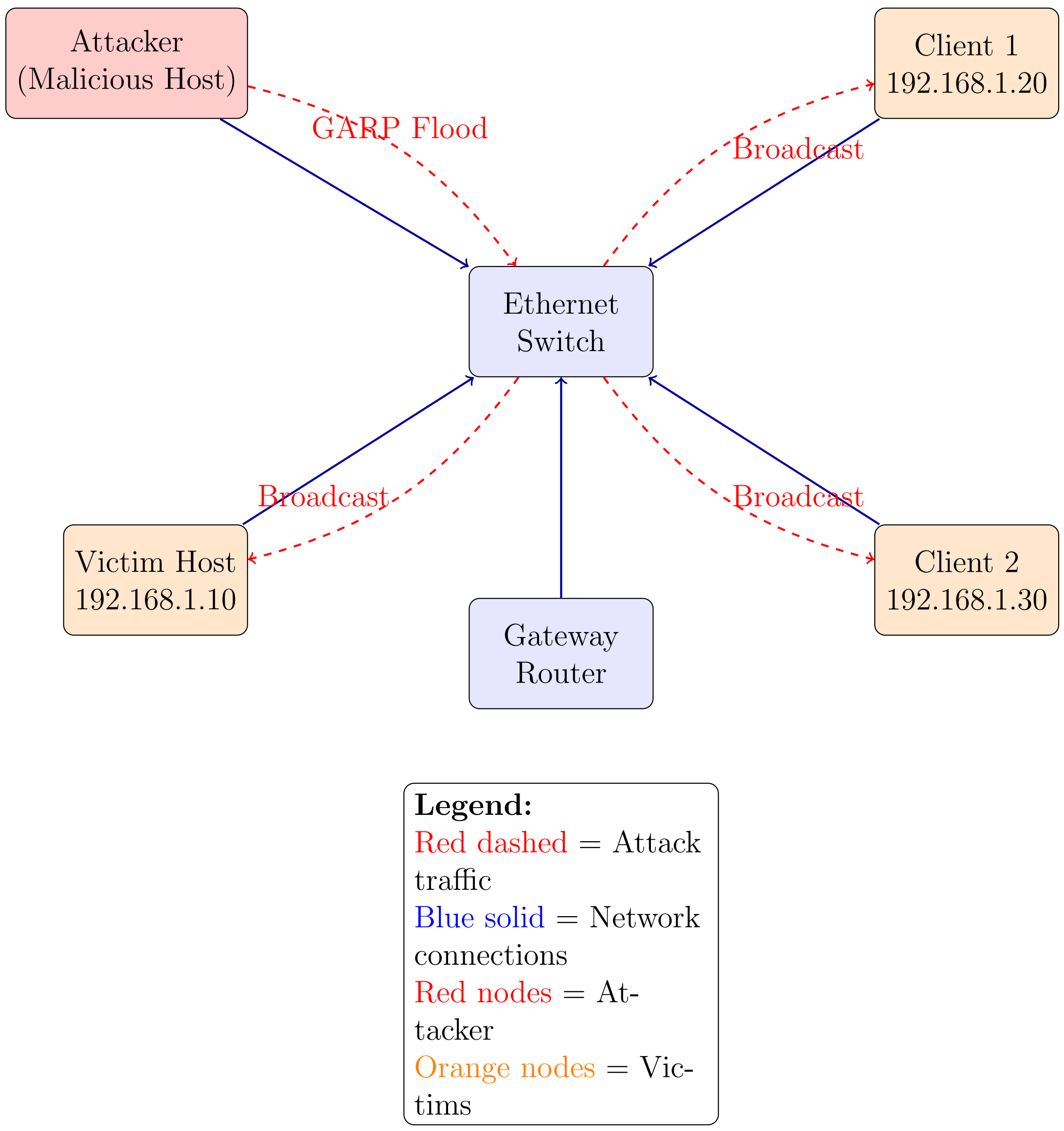


Figure 2: Network Topology for Gratuitous ARP Storm Attack

4 Attack Implementation Analysis

4.1 Gratuitous ARP Storm Attack

4.1.1 Technical Implementation

The attack is implemented through custom packet crafting:

Ethernet Frame Structure (14 bytes):

- Destination MAC (6 bytes): Broadcast address (FF:FF:FF:FF:FF:FF)
- Source MAC (6 bytes): Randomized attacker MAC
- EtherType (2 bytes): 0x0806 (ARP protocol identifier)

ARP Packet Structure (28 bytes):

- Hardware Type: 0x0001 (Ethernet)
- Protocol Type: 0x0800 (IPv4)
- Hardware Length: 6 (MAC address length)
- Protocol Length: 4 (IP address length)
- Operation: 0x0002 (ARP Reply)
- Sender Hardware/Protocol Addresses: Spoofed values
- Target Hardware/Protocol Addresses: Victim information

4.1.2 Attack Mechanism

1. **Packet Generation**: Creates malicious ARP packets with fake IP-MAC mappings
2. **Network Flooding**: Sends thousands of packets per second to overwhelm infrastructure
3. **Cache Poisoning**: Fills ARP caches with false entries
4. **Resource Exhaustion**: Consumes network bandwidth and processing power
5. **Service Disruption**: Causes network devices to become unresponsive

4.2 Targeted ARP Poisoning

The targeted poisoning attack implements man-in-the-middle capabilities:

- Sends forged ARP replies to victim claiming to be the gateway
- Sends forged ARP replies to gateway claiming to be the victim
- Enables traffic interception and manipulation
- Supports bi-directional MITM positioning

4.3 Performance Capabilities

Table 6: Attack Tool Performance Metrics

Implementation	Packets/Second	Threading	Memory Usage
Python Tool	1,000-5,000	Multi-threaded	<50MB

5 Defense and Detection Mechanisms

5.1 Real-Time Detection System

The project implements comprehensive detection mechanisms through the Observer container:

5.1.1 Detection Thresholds

Table 7: Attack Detection Thresholds

Metric	Threshold	Attack Indicator
ARP packets/second	>50	Network flooding
Unique MACs/minute	>20	Distributed attack
Gratuitous ARP ratio	>70%	ARP spoofing
MAC changes per IP (5 min)	>3	IP spoofing

5.1.2 Detection Algorithms

1. **Packet Rate Analysis**: Monitors ARP packet frequency using sliding time windows
2. **Sender Diversity Tracking**: Counts unique MAC addresses within time intervals
3. **Gratuitous ARP Detection**: Identifies unsolicited ARP replies
4. **IP-MAC Consistency Monitoring**: Tracks changes in IP-to-MAC mappings
5. **Anomaly Correlation**: Combines multiple indicators for attack assessment

5.2 Mitigation Strategies

there can be several defense mechanisms:

5.2.1 Network-Level Defenses

- **Rate Limiting**: Implement ARP packet processing limits
- **Static ARP Entries**: Use fixed IP-MAC mappings for critical devices
- **ARP Inspection**: Validate ARP packet sources and legitimacy
- **Network Segmentation**: Isolate broadcast domains

5.2.2 Example Implementation

Listing 1: Linux ARP Rate Limiting Implementation

```
# Rate limit ARP packets to 10 per second
iptables -A INPUT -p arp -m limit --limit 10/sec -j ACCEPT
iptables -A INPUT -p arp -j DROP

# Set static ARP entry for critical gateway
arp -s 192.168.1.1 00:11:22:33:44:55
```

5.3 Monitoring and Analysis Framework

5.3.1 Real-Time Monitoring

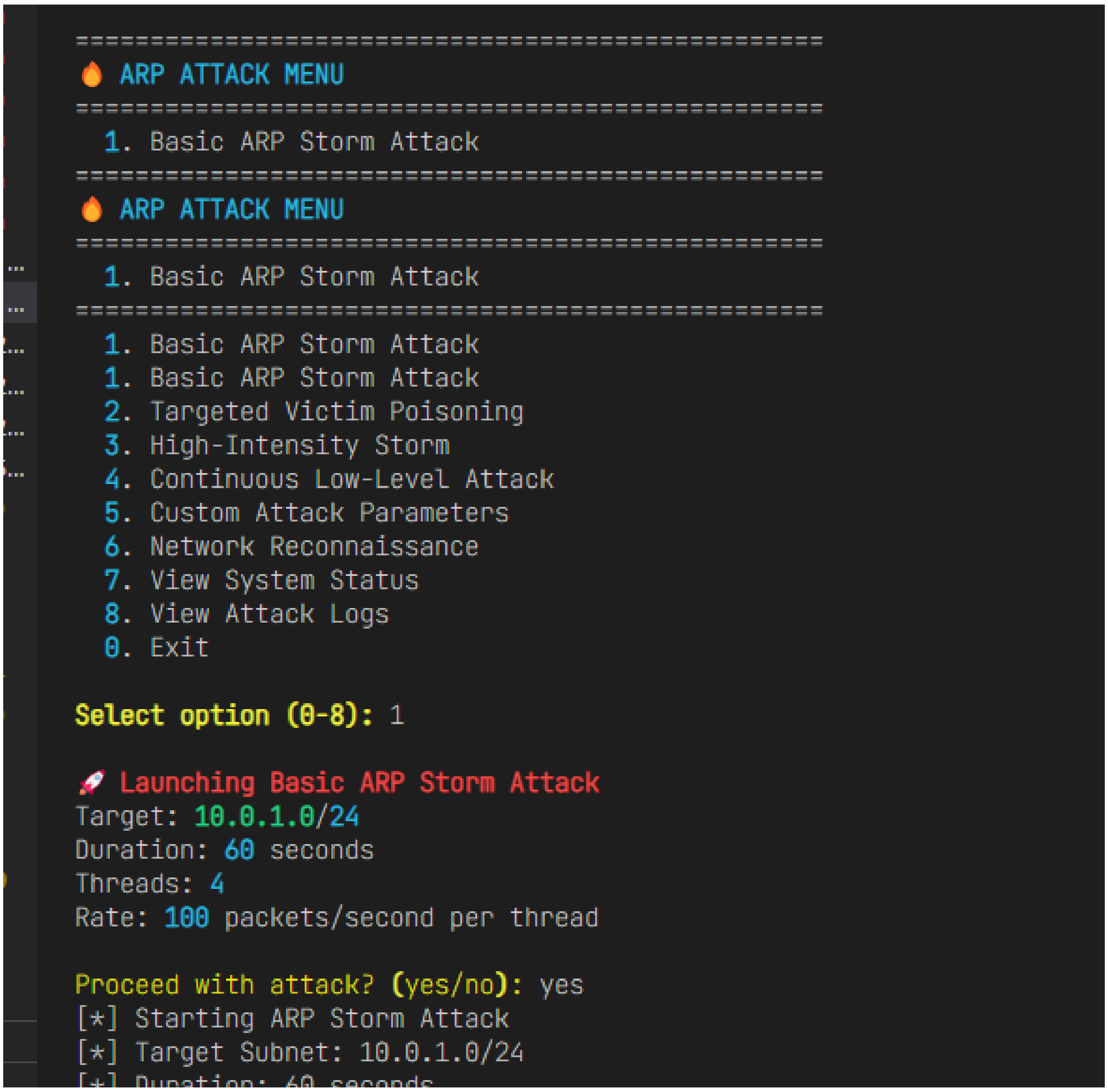
- Web dashboard at localhost:8080 for visual monitoring
- Real-time traffic graphs and statistics
- Container status and health monitoring
- Attack visualization with charts and metrics

5.3.2 Traffic Analysis Capabilities

- Packet capture to .pcap files for forensic analysis
- JSON report generation with attack statistics
- Attack pattern identification and classification
- Performance metrics and effectiveness measurement

6 Steps of the attack

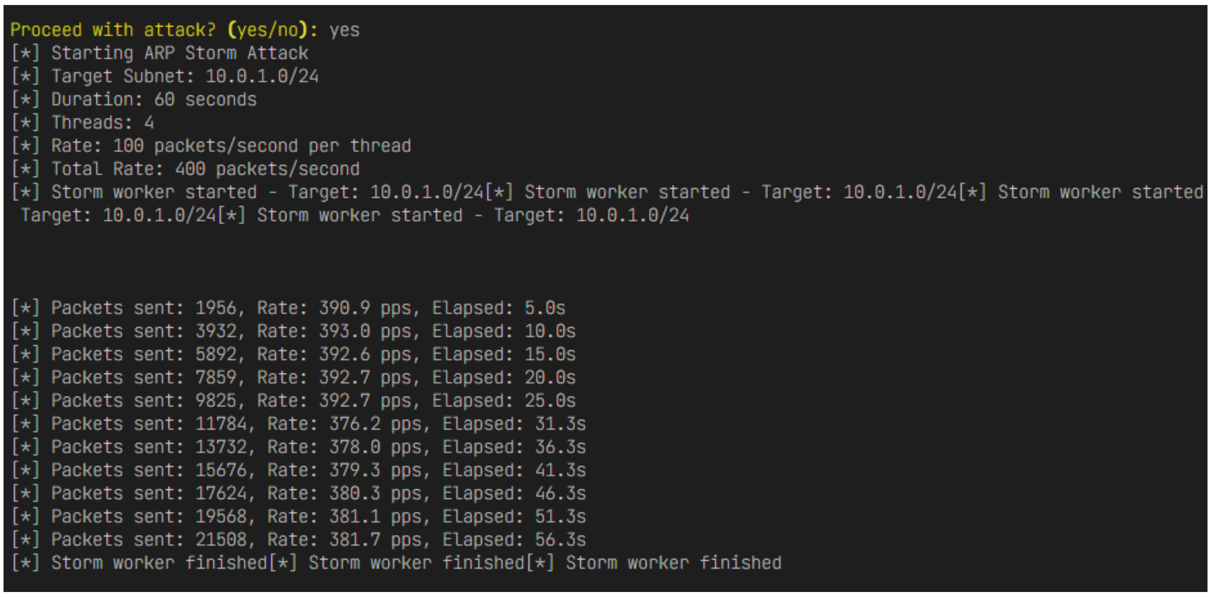
We first run the all of the above observer, atatcker victim in theh docker virtual network. THen we command into attacker code, via docker, the docker then executed the attack, and we get to see a menu



```
==================================================
🔥 ARP ATTACK MENU
==================================================
  1. Basic ARP Storm Attack
==================================================
🔥 ARP ATTACK MENU
==================================================
  1. Basic ARP Storm Attack
==================================================
  1. Basic ARP Storm Attack
  1. Basic ARP Storm Attack
  2. Targeted Victim Poisoning
  3. High-Intensity Storm
  4. Continuous Low-Level Attack
  5. Custom Attack Parameters
  6. Network Reconnaissance
  7. View System Status
  8. View Attack Logs
  0. Exit

Select option (0-8): 1

🚀 Launching Basic ARP Storm Attack
Target: 10.0.1.0/24
Duration: 60 seconds
Threads: 4
Rate: 100 packets/second per thread

Proceed with attack? (yes/no): yes
[*] Starting ARP Storm Attack
[*] Target Subnet: 10.0.1.0/24
[*] Duration: 60 seconds
```

Figure 3: Attack Menu

Now by selecting the ARP storm option, what it does is it creates 4 threads to attack the victim, it also takes the victim IP address from the environment variable, and then starts the storm attack, that means it selects a time frame for the attack (like 60 seconds) and then starts each of threads. Now the flag running also set to true. After that when the running the flag is on and the total time of the attack is still less than the alloted time, it will continue to print the packets sent in every 5 seconds to log in the command line.



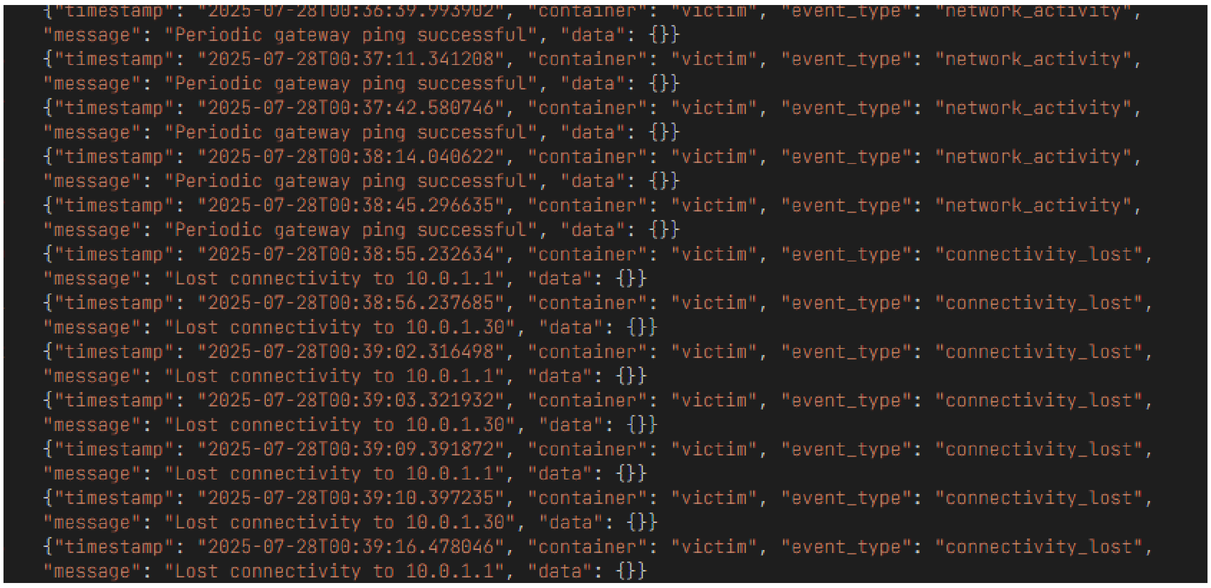
```
Proceed with attack? (yes/no): yes
[*] Starting ARP Storm Attack
[*] Target Subnet: 10.0.1.0/24
[*] Duration: 60 seconds
[*] Threads: 4
[*] Rate: 100 packets/second per thread
[*] Total Rate: 400 packets/second
[*] Storm worker started - Target: 10.0.1.0/24[*] Storm worker started - Target: 10.0.1.0/24[*] Storm worker started
 Target: 10.0.1.0/24[*] Storm worker started - Target: 10.0.1.0/24



[*] Packets sent: 1956, Rate: 390.9 pps, Elapsed: 5.0s
[*] Packets sent: 3932, Rate: 393.0 pps, Elapsed: 10.0s
[*] Packets sent: 5892, Rate: 392.6 pps, Elapsed: 15.0s
[*] Packets sent: 7859, Rate: 392.7 pps, Elapsed: 20.0s
[*] Packets sent: 9825, Rate: 392.7 pps, Elapsed: 25.0s
[*] Packets sent: 11784, Rate: 376.2 pps, Elapsed: 31.3s
[*] Packets sent: 13732, Rate: 378.0 pps, Elapsed: 36.3s
[*] Packets sent: 15676, Rate: 379.3 pps, Elapsed: 41.3s
[*] Packets sent: 17624, Rate: 380.3 pps, Elapsed: 46.3s
[*] Packets sent: 19568, Rate: 381.1 pps, Elapsed: 51.3s
[*] Packets sent: 21508, Rate: 381.7 pps, Elapsed: 56.3s
[*] Storm worker finished[*] Storm worker finished[*] Storm worker finished
```

Figure 4: Attacker logging the number of packets sent

What it actually does for storm attack is that is creates a raw socket and set it up to work with raw ethernet frames. And then it sets the interval to maintain the packet spreading rate. And then it continues to run until the end of the time frame. IN each time it sends random source mac and random IP addresses. It creates gratuitous arp packets and then send it on a frame. Its goal is to Flood the network with fake ARP packets to disrupt communication (Denial-of-Service attack). Overwhelm switches and hosts with excessive ARP traffic, causing Network Slowdowns or crashes. As a result, Switches get overwhelmed, leading to MAC table flooding (some switches fail open, broadcasting all traffic). Legitimate ARP requests being dropped. Thus the hosts experience Intermittent connectivity loss (incorrect ARP cache entries). Also increased CPU usage due to processing fake ARP packets.

7 Is the Attack Successful?

Yes, the attack is successful. The victim and observer continuously provide log files to show any kind of attack. In the victim logs we can see that

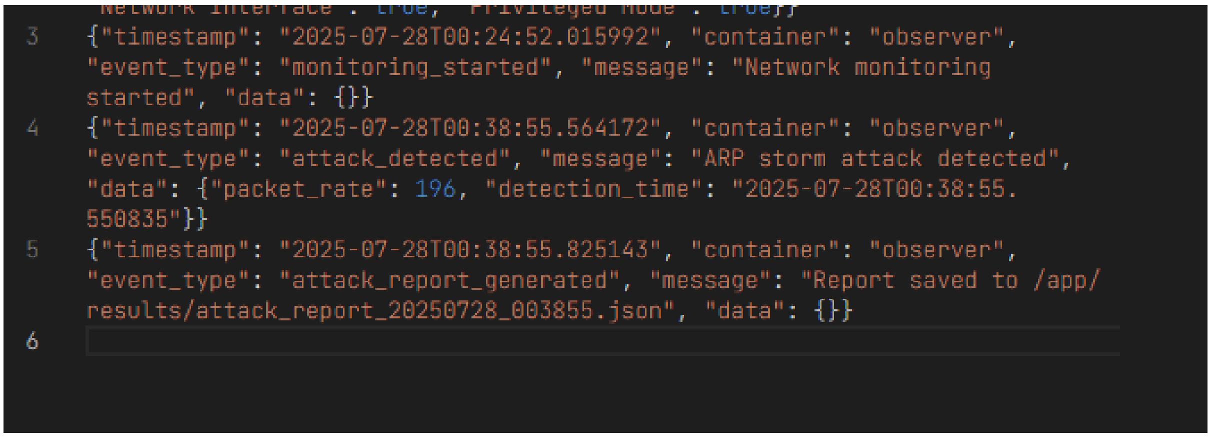


```
{"timestamp": "2025-07-28T00:36:39.993902", "container": "victim", "event_type": "network_activity",
"message": "Periodic gateway ping successful", "data": {}}
{"timestamp": "2025-07-28T00:37:11.341208", "container": "victim", "event_type": "network_activity",
"message": "Periodic gateway ping successful", "data": {}}
{"timestamp": "2025-07-28T00:37:42.580746", "container": "victim", "event_type": "network_activity",
"message": "Periodic gateway ping successful", "data": {}}
{"timestamp": "2025-07-28T00:38:14.040622", "container": "victim", "event_type": "network_activity",
"message": "Periodic gateway ping successful", "data": {}}
{"timestamp": "2025-07-28T00:38:45.296635", "container": "victim", "event_type": "network_activity",
"message": "Periodic gateway ping successful", "data": {}}
{"timestamp": "2025-07-28T00:38:55.232634", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.1", "data": {}}
{"timestamp": "2025-07-28T00:38:56.237685", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.30", "data": {}}
{"timestamp": "2025-07-28T00:39:02.316498", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.1", "data": {}}
{"timestamp": "2025-07-28T00:39:03.321932", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.30", "data": {}}
{"timestamp": "2025-07-28T00:39:09.391872", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.1", "data": {}}
{"timestamp": "2025-07-28T00:39:10.397235", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.30", "data": {}}
{"timestamp": "2025-07-28T00:39:16.478046", "container": "victim", "event_type": "connectivity_lost",
"message": "Lost connectivity to 10.0.1.1", "data": {}}
```

Figure 5: Logs of victim

As soon as we started the attack, the victim lost its network connectivity to both the gateway and the observer.

The observer also found the attack



```
"Network Interface": true, "Privileged Mode": true}}
{"timestamp": "2025-07-28T00:24:52.015992", "container": "observer",
"event_type": "monitoring_started", "message": "Network monitoring
started", "data": {}}
{"timestamp": "2025-07-28T00:38:55.564172", "container": "observer",
"event_type": "attack_detected", "message": "ARP storm attack detected",
"data": {"packet_rate": 196, "detection_time": "2025-07-28T00:38:55.
550835"}}
{"timestamp": "2025-07-28T00:38:55.825143", "container": "observer",
"event_type": "attack_report_generated", "message": "Report saved to /app/
results/attack_report_20250728_003855.json", "data": {}}
```

Figure 6: Logs of observer

Now the report was generated during the visualization time.

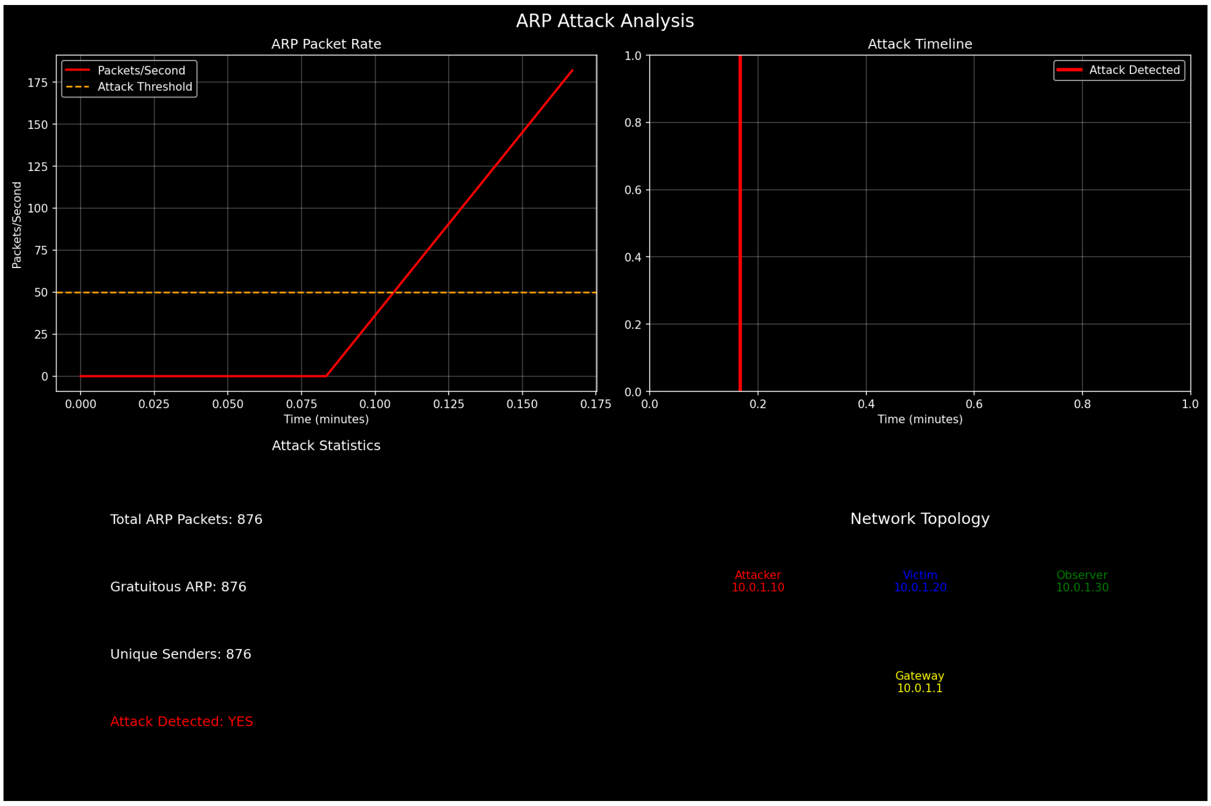
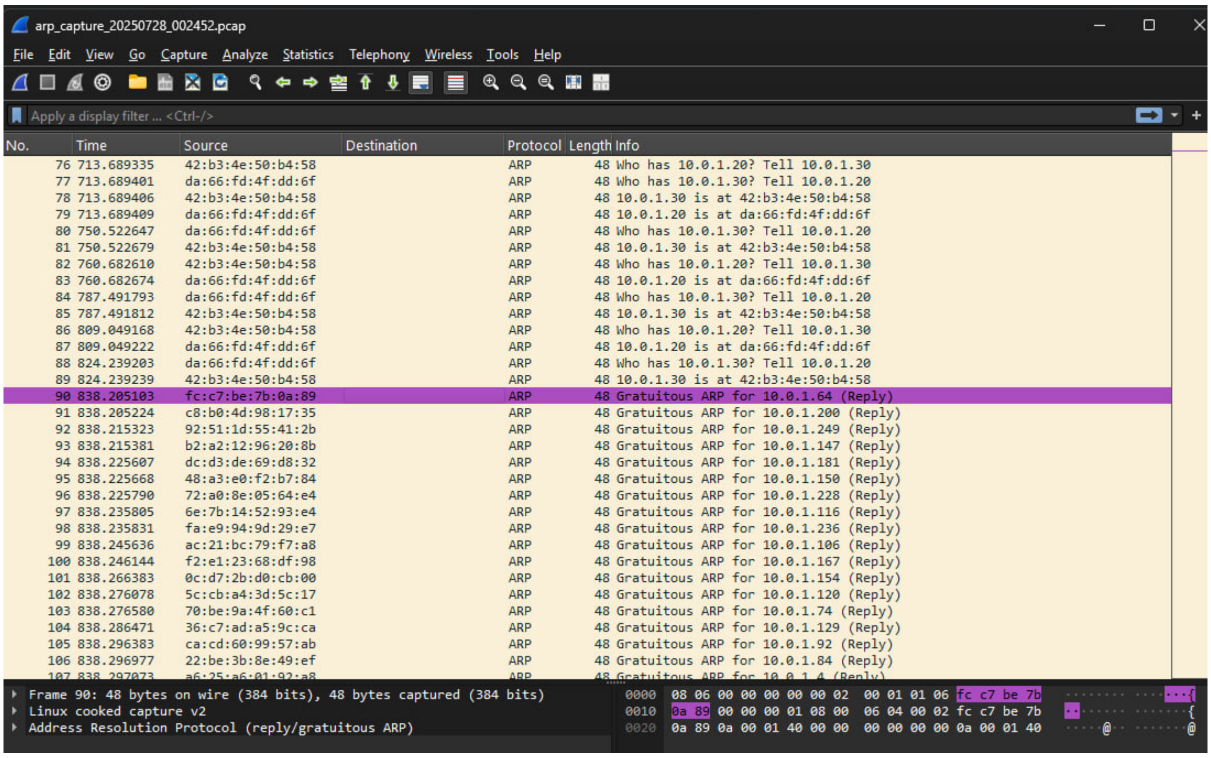


Figure 7: Attack Graph (ARP Packets in a minute

Also the interaction saved into pcap file where intially we can see that



arp_capture_20250728_002452.pcap

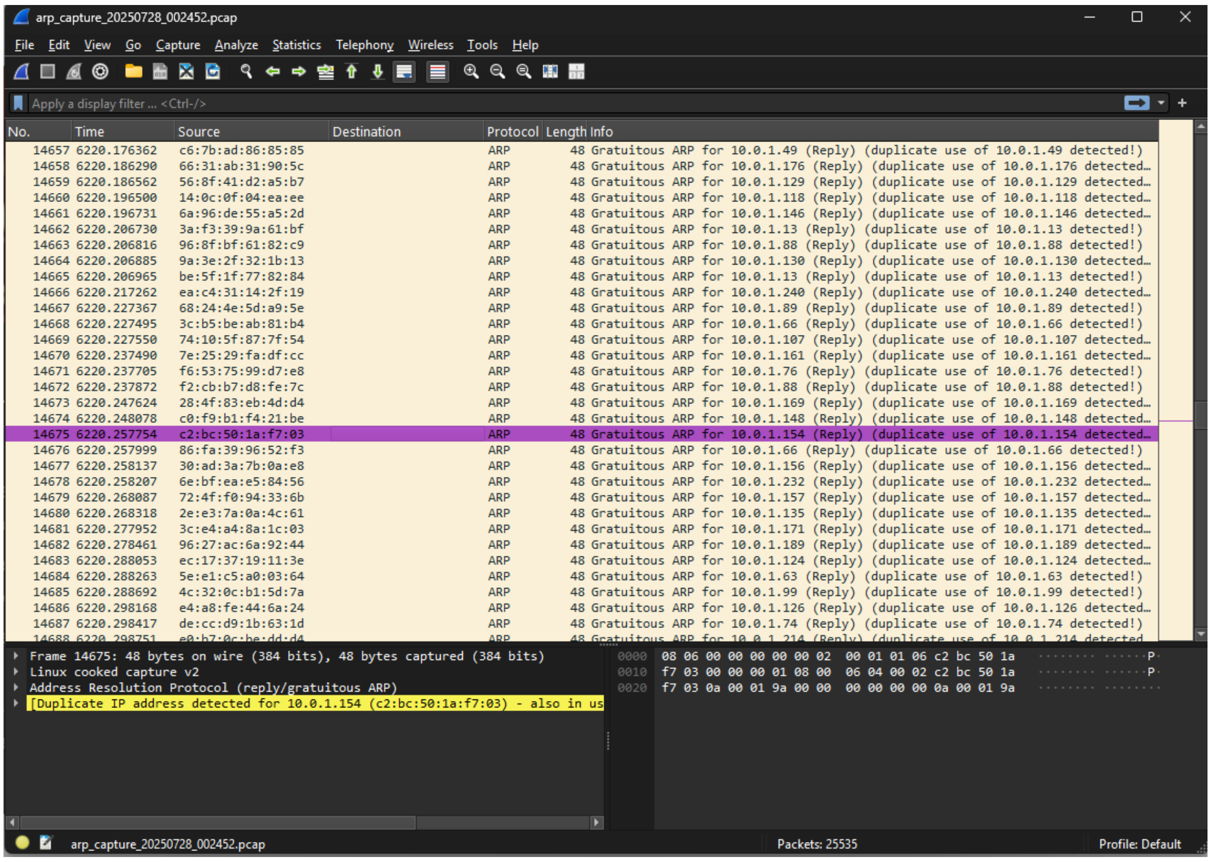
```
No.  Time        Source             Destination  Protocol Length Info
 76 713.689335  42:b3:4e:50:b4:58               ARP      48 Who has 10.0.1.20? Tell 10.0.1.30
 77 713.689401  da:66:fd:4f:dd:6f               ARP      48 Who has 10.0.1.30? Tell 10.0.1.20
 78 713.689406  42:b3:4e:50:b4:58               ARP      48 10.0.1.30 is at 42:b3:4e:50:b4:58
 79 713.689409  da:66:fd:4f:dd:6f               ARP      48 10.0.1.20 is at da:66:fd:4f:dd:6f
 80 750.522647  da:66:fd:4f:dd:6f               ARP      48 Who has 10.0.1.30? Tell 10.0.1.20
 81 750.522679  42:b3:4e:50:b4:58               ARP      48 10.0.1.30 is at 42:b3:4e:50:b4:58
 82 760.682610  42:b3:4e:50:b4:58               ARP      48 Who has 10.0.1.20? Tell 10.0.1.30
 83 760.682674  da:66:fd:4f:dd:6f               ARP      48 10.0.1.20 is at da:66:fd:4f:dd:6f
 84 787.491793  da:66:fd:4f:dd:6f               ARP      48 Who has 10.0.1.30? Tell 10.0.1.20
 85 787.491812  42:b3:4e:50:b4:58               ARP      48 10.0.1.30 is at 42:b3:4e:50:b4:58
 86 809.049168  42:b3:4e:50:b4:58               ARP      48 Who has 10.0.1.20? Tell 10.0.1.30
 87 809.049222  da:66:fd:4f:dd:6f               ARP      48 10.0.1.20 is at da:66:fd:4f:dd:6f
 88 824.239203  da:66:fd:4f:dd:6f               ARP      48 Who has 10.0.1.30? Tell 10.0.1.20
 89 824.239239  42:b3:4e:50:b4:58               ARP      48 10.0.1.30 is at 42:b3:4e:50:b4:58
 90 838.205103  fc:c7:be:7b:0a:89               ARP      48 Gratuitous ARP for 10.0.1.64 (Reply)
 91 838.205224  c8:b0:4d:98:17:35               ARP      48 Gratuitous ARP for 10.0.1.200 (Reply)
 92 838.215323  92:51:1d:55:41:2b               ARP      48 Gratuitous ARP for 10.0.1.249 (Reply)
 93 838.215381  b2:a2:12:96:20:8b               ARP      48 Gratuitous ARP for 10.0.1.147 (Reply)
 94 838.225607  dc:d3:de:69:d8:32               ARP      48 Gratuitous ARP for 10.0.1.181 (Reply)
 95 838.225668  48:a3:e0:f2:b7:84               ARP      48 Gratuitous ARP for 10.0.1.150 (Reply)
 96 838.225790  72:a0:8e:05:64:e4               ARP      48 Gratuitous ARP for 10.0.1.228 (Reply)
 97 838.235805  6e:7b:14:52:93:e4               ARP      48 Gratuitous ARP for 10.0.1.116 (Reply)
 98 838.235831  fa:e9:94:9d:29:e7               ARP      48 Gratuitous ARP for 10.0.1.236 (Reply)
 99 838.245636  ac:21:bc:79:f7:a8               ARP      48 Gratuitous ARP for 10.0.1.106 (Reply)
100 838.246144  f2:e1:23:68:df:98               ARP      48 Gratuitous ARP for 10.0.1.167 (Reply)
101 838.266383  0c:d7:2b:d0:cb:00               ARP      48 Gratuitous ARP for 10.0.1.154 (Reply)
102 838.276078  5c:cb:a4:3d:5c:17               ARP      48 Gratuitous ARP for 10.0.1.120 (Reply)
103 838.276580  70:be:9a:4f:60:c1               ARP      48 Gratuitous ARP for 10.0.1.74 (Reply)
104 838.286471  36:c7:ad:a5:9c:ca               ARP      48 Gratuitous ARP for 10.0.1.129 (Reply)
105 838.296383  ca:cd:60:99:57:ab               ARP      48 Gratuitous ARP for 10.0.1.92 (Reply)
106 838.296977  22:be:3b:8e:49:ef               ARP      48 Gratuitous ARP for 10.0.1.84 (Reply)
107 838.297073  a6:25:a6:01:92:a8               ARP      48 Gratuitous ARP for 10.0.1.4 (Reply)
```

```
> Frame 90: 48 bytes on wire (384 bits), 48 bytes captured (384 bits)
> Linux cooked capture v2
> Address Resolution Protocol (reply/gratuitous ARP)
```

```
0000  08 06 00 00 00 00 00 02  00 01 01 06 fc c7 be 7b   ········ ·······{
0010  0a 89 00 00 00 01 08 00  06 04 00 02 fc c7 be 7b   ········ ·······{
0020  0a 89 0a 00 01 40 00 00  00 00 00 00 0a 00 01 40   ·····@·· ·······@
```

Figure 8: Initialyl Pcap

But as the attck continues we then get to see that duplicate gratuitous ARP packets are being found


arp_capture_20250728_002452.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time        Source             Destination  Protocol Length Info
14657 6220.176362 c6:7b:ad:86:85:85               ARP      48 Gratuitous ARP for 10.0.1.49 (Reply) (duplicate use of 10.0.1.49 detected!)
14658 6220.186290 66:31:ab:31:90:5c               ARP      48 Gratuitous ARP for 10.0.1.176 (Reply) (duplicate use of 10.0.1.176 detected…
14659 6220.186562 56:8f:41:d2:a5:b7               ARP      48 Gratuitous ARP for 10.0.1.129 (Reply) (duplicate use of 10.0.1.129 detected…
14660 6220.196500 14:0c:0f:04:ea:ee               ARP      48 Gratuitous ARP for 10.0.1.118 (Reply) (duplicate use of 10.0.1.118 detected…
14661 6220.196731 6a:96:de:55:a5:2d               ARP      48 Gratuitous ARP for 10.0.1.146 (Reply) (duplicate use of 10.0.1.146 detected…
14662 6220.206730 3a:f3:39:9a:61:bf               ARP      48 Gratuitous ARP for 10.0.1.13 (Reply) (duplicate use of 10.0.1.13 detected!)
14663 6220.206816 96:8f:bf:61:82:c9               ARP      48 Gratuitous ARP for 10.0.1.88 (Reply) (duplicate use of 10.0.1.88 detected!)
14664 6220.206885 9a:3e:2f:32:1b:13               ARP      48 Gratuitous ARP for 10.0.1.130 (Reply) (duplicate use of 10.0.1.130 detected…
14665 6220.206965 be:5f:1f:77:82:84               ARP      48 Gratuitous ARP for 10.0.1.13 (Reply) (duplicate use of 10.0.1.13 detected!)
14666 6220.217262 ea:c4:31:14:2f:19               ARP      48 Gratuitous ARP for 10.0.1.240 (Reply) (duplicate use of 10.0.1.240 detected…
14667 6220.227367 68:24:4e:5d:a9:5e               ARP      48 Gratuitous ARP for 10.0.1.89 (Reply) (duplicate use of 10.0.1.89 detected!)
14668 6220.227495 3c:b5:be:ab:81:b4               ARP      48 Gratuitous ARP for 10.0.1.66 (Reply) (duplicate use of 10.0.1.66 detected!)
14669 6220.227550 74:10:5f:87:7f:54               ARP      48 Gratuitous ARP for 10.0.1.107 (Reply) (duplicate use of 10.0.1.107 detected…
14670 6220.237490 7e:25:29:fa:df:cc               ARP      48 Gratuitous ARP for 10.0.1.161 (Reply) (duplicate use of 10.0.1.161 detected…
14671 6220.237705 f6:53:75:99:d7:e8               ARP      48 Gratuitous ARP for 10.0.1.76 (Reply) (duplicate use of 10.0.1.76 detected!)
14672 6220.237872 f2:cb:b7:d8:fe:7c               ARP      48 Gratuitous ARP for 10.0.1.88 (Reply) (duplicate use of 10.0.1.88 detected!)
14673 6220.247624 28:4f:83:eb:4d:d4               ARP      48 Gratuitous ARP for 10.0.1.169 (Reply) (duplicate use of 10.0.1.169 detected…
14674 6220.248078 c0:f9:b1:f4:21:be               ARP      48 Gratuitous ARP for 10.0.1.148 (Reply) (duplicate use of 10.0.1.148 detected…
14675 6220.257754 c2:bc:50:1a:f7:03               ARP      48 Gratuitous ARP for 10.0.1.154 (Reply) (duplicate use of 10.0.1.154 detected…
14676 6220.257999 86:fa:39:96:52:f3               ARP      48 Gratuitous ARP for 10.0.1.66 (Reply) (duplicate use of 10.0.1.66 detected!)
14677 6220.258137 30:ad:3a:7b:0a:e8               ARP      48 Gratuitous ARP for 10.0.1.156 (Reply) (duplicate use of 10.0.1.156 detected…
14678 6220.258207 6e:bf:ea:e5:84:56               ARP      48 Gratuitous ARP for 10.0.1.232 (Reply) (duplicate use of 10.0.1.232 detected…
14679 6220.268087 72:4f:f0:94:33:6b               ARP      48 Gratuitous ARP for 10.0.1.157 (Reply) (duplicate use of 10.0.1.157 detected…
14680 6220.268318 2e:e3:7a:0a:4c:61               ARP      48 Gratuitous ARP for 10.0.1.135 (Reply) (duplicate use of 10.0.1.135 detected…
14681 6220.277952 3c:e4:a4:8a:1c:03               ARP      48 Gratuitous ARP for 10.0.1.171 (Reply) (duplicate use of 10.0.1.171 detected…
14682 6220.278461 96:27:ac:6a:92:44               ARP      48 Gratuitous ARP for 10.0.1.189 (Reply) (duplicate use of 10.0.1.189 detected…
14683 6220.288053 ec:17:37:19:11:3e               ARP      48 Gratuitous ARP for 10.0.1.124 (Reply) (duplicate use of 10.0.1.124 detected…
14684 6220.288263 5e:e1:c5:a0:03:64               ARP      48 Gratuitous ARP for 10.0.1.63 (Reply) (duplicate use of 10.0.1.63 detected!)
14685 6220.288692 4c:32:0c:b1:5d:7a               ARP      48 Gratuitous ARP for 10.0.1.99 (Reply) (duplicate use of 10.0.1.99 detected!)
14686 6220.298168 e4:a8:fe:44:6a:24               ARP      48 Gratuitous ARP for 10.0.1.126 (Reply) (duplicate use of 10.0.1.126 detected…
14687 6220.298417 de:cc:d9:1b:63:1d               ARP      48 Gratuitous ARP for 10.0.1.74 (Reply) (duplicate use of 10.0.1.74 detected!)
14688 6220.298751 e0:b7:0c:be:dd:d4               ARP      48 Gratuitous ARP for 10.0.1.214 (Reply) (duplicate use of 10.0.1.214 detected
```

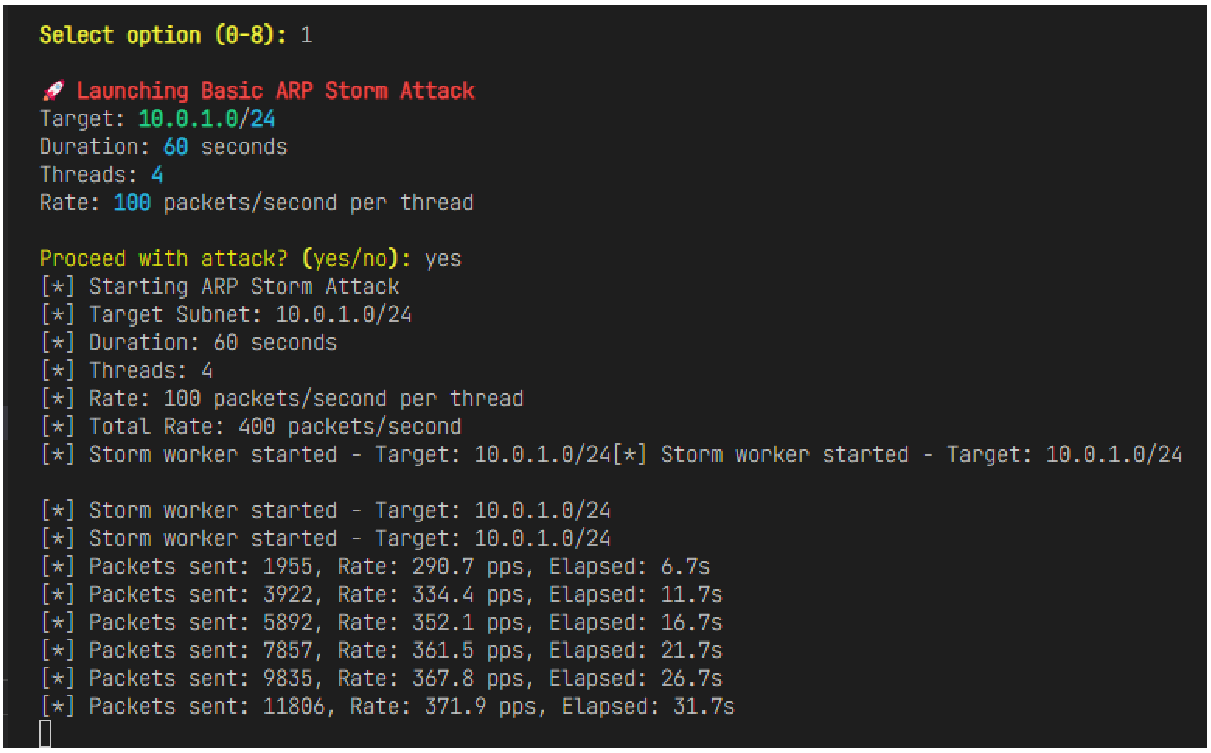
```
Frame 14675: 48 bytes on wire (384 bits), 48 bytes captured (384 bits)
Linux cooked capture v2
Address Resolution Protocol (reply/gratuitous ARP)
[Duplicate IP address detected for 10.0.1.154 (c2:bc:50:1a:f7:03) - also in us
```

```
0000  08 06 00 00 00 00 00 02  00 01 01 06 c2 bc 50 1a   ········ ······P·
0010  f7 03 00 00 00 01 08 00  06 04 00 02 c2 bc 50 1a   ········ ······P·
0020  f7 03 0a 00 01 9a 00 00  00 00 00 00 0a 00 01 9a   ········ ········
```

Figure 9: Duplicate ARP gratuitous packets in pcap

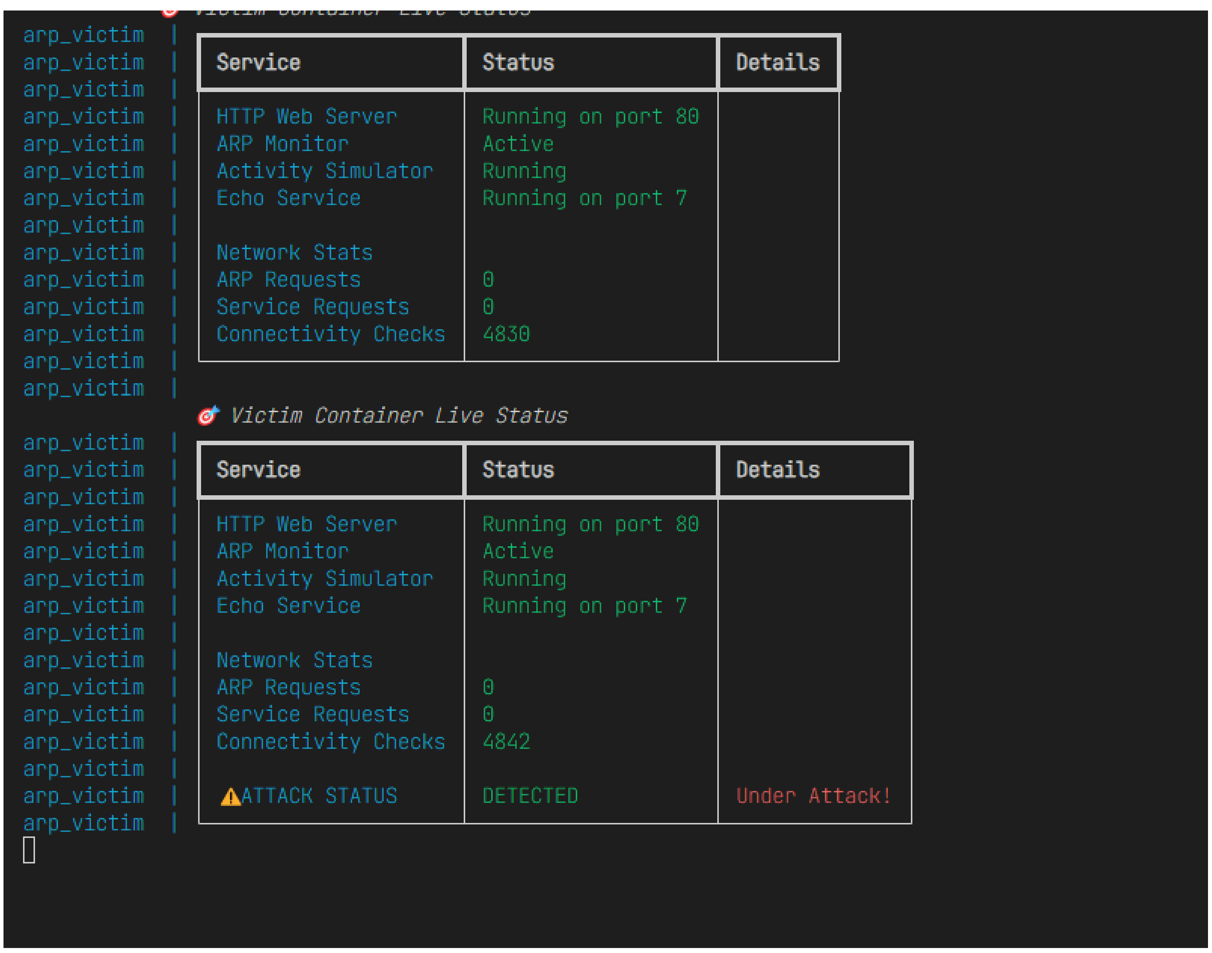
So, the attack was successful of creating an ARP storm and Denial of Service.

8 Observed Output in different PCs



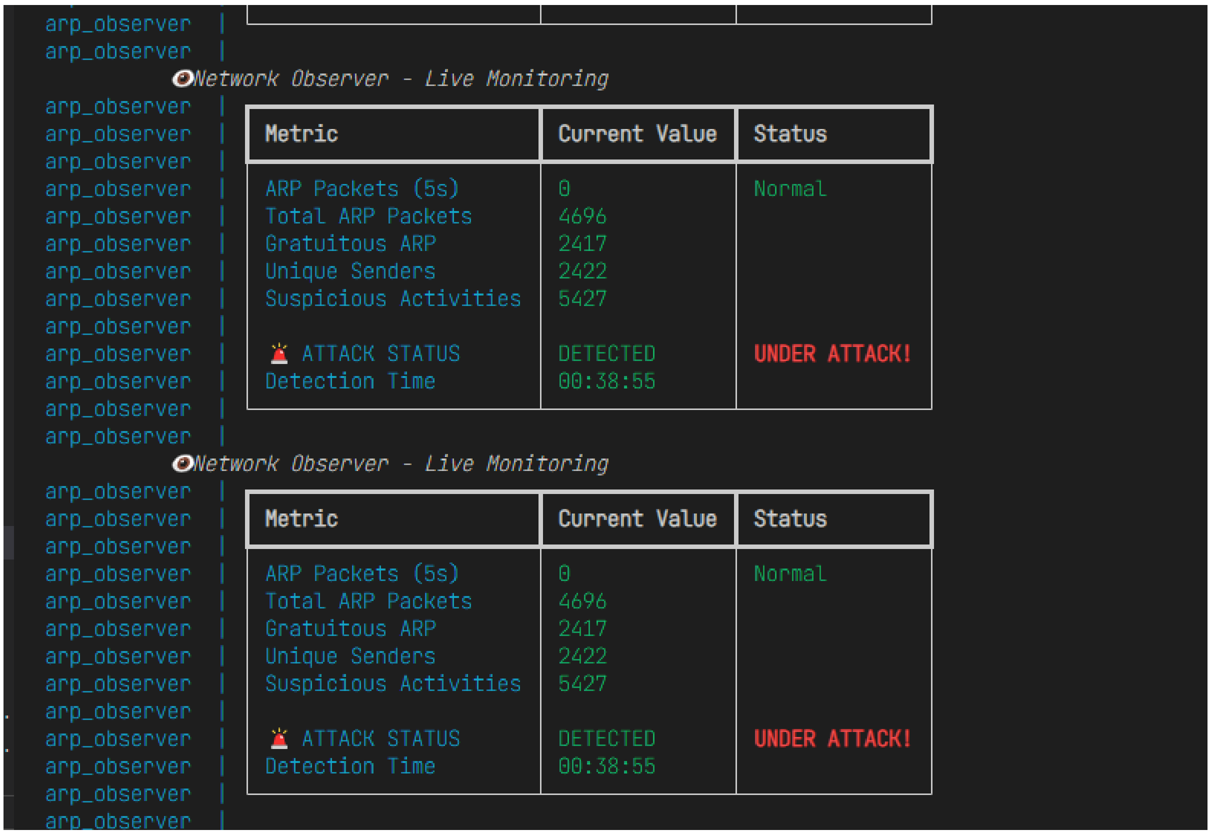
```
Select option (0-8): 1

🚀 Launching Basic ARP Storm Attack
Target: 10.0.1.0/24
Duration: 60 seconds
Threads: 4
Rate: 100 packets/second per thread

Proceed with attack? (yes/no): yes
[*] Starting ARP Storm Attack
[*] Target Subnet: 10.0.1.0/24
[*] Duration: 60 seconds
[*] Threads: 4
[*] Rate: 100 packets/second per thread
[*] Total Rate: 400 packets/second
[*] Storm worker started - Target: 10.0.1.0/24[*] Storm worker started - Target: 10.0.1.0/24

[*] Storm worker started - Target: 10.0.1.0/24
[*] Storm worker started - Target: 10.0.1.0/24
[*] Packets sent: 1955, Rate: 290.7 pps, Elapsed: 6.7s
[*] Packets sent: 3922, Rate: 334.4 pps, Elapsed: 11.7s
[*] Packets sent: 5892, Rate: 352.1 pps, Elapsed: 16.7s
[*] Packets sent: 7857, Rate: 361.5 pps, Elapsed: 21.7s
[*] Packets sent: 9835, Rate: 367.8 pps, Elapsed: 26.7s
[*] Packets sent: 11806, Rate: 371.9 pps, Elapsed: 31.7s
```

Figure 10: Output at attacker



```
arp_victim  |  ┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━┓
arp_victim  |  ┃ Service             ┃ Status             ┃ Details ┃
arp_victim  |  ┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━┩
arp_victim  |  │ HTTP Web Server     │ Running on port 80 │         │
arp_victim  |  │ ARP Monitor         │ Active             │         │
arp_victim  |  │ Activity Simulator  │ Running            │         │
arp_victim  |  │ Echo Service        │ Running on port 7  │         │
arp_victim  |  │                     │                    │         │
arp_victim  |  │ Network Stats       │                    │         │
arp_victim  |  │ ARP Requests        │ 0                  │         │
arp_victim  |  │ Service Requests    │ 0                  │         │
arp_victim  |  │ Connectivity Checks │ 4830               │         │
arp_victim  |  └─────────────────────┴────────────────────┴─────────┘
arp_victim  |
               🎯 Victim Container Live Status
arp_victim  |  ┏━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
arp_victim  |  ┃ Service             ┃ Status             ┃ Details       ┃
arp_victim  |  ┡━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
arp_victim  |  │ HTTP Web Server     │ Running on port 80 │               │
arp_victim  |  │ ARP Monitor         │ Active             │               │
arp_victim  |  │ Activity Simulator  │ Running            │               │
arp_victim  |  │ Echo Service        │ Running on port 7  │               │
arp_victim  |  │                     │                    │               │
arp_victim  |  │ Network Stats       │                    │               │
arp_victim  |  │ ARP Requests        │ 0                  │               │
arp_victim  |  │ Service Requests    │ 0                  │               │
arp_victim  |  │ Connectivity Checks │ 4842               │               │
arp_victim  |  │                     │                    │               │
arp_victim  |  │ ⚠ATTACK STATUS      │ DETECTED           │ Under Attack! │
arp_victim  |  └─────────────────────┴────────────────────┴───────────────┘
```

Figure 11: Output at victim PC



```
arp_observer  |
arp_observer  |
                 Network Observer - Live Monitoring
arp_observer  |  ┌────────────────────┬────────────────┬───────────────┐
arp_observer  |  │ Metric             │ Current Value  │ Status        │
arp_observer  |  ├────────────────────┼────────────────┼───────────────┤
arp_observer  |  │ ARP Packets (5s)   │ 0              │ Normal        │
arp_observer  |  │ Total ARP Packets  │ 4696           │               │
arp_observer  |  │ Gratuitous ARP     │ 2417           │               │
arp_observer  |  │ Unique Senders     │ 2422           │               │
arp_observer  |  │ Suspicious Activities │ 5427        │               │
arp_observer  |  │                    │                │               │
arp_observer  |  │ 🚨 ATTACK STATUS   │ DETECTED       │ UNDER ATTACK! │
arp_observer  |  │ Detection Time     │ 00:38:55       │               │
arp_observer  |  └────────────────────┴────────────────┴───────────────┘
arp_observer  |
                 Network Observer - Live Monitoring
arp_observer  |  ┌────────────────────┬────────────────┬───────────────┐
arp_observer  |  │ Metric             │ Current Value  │ Status        │
arp_observer  |  ├────────────────────┼────────────────┼───────────────┤
arp_observer  |  │ ARP Packets (5s)   │ 0              │ Normal        │
arp_observer  |  │ Total ARP Packets  │ 4696           │               │
arp_observer  |  │ Gratuitous ARP     │ 2417           │               │
arp_observer  |  │ Unique Senders     │ 2422           │               │
arp_observer  |  │ Suspicious Activities │ 5427        │               │
arp_observer  |  │                    │                │               │
arp_observer  |  │ 🚨 ATTACK STATUS   │ DETECTED       │ UNDER ATTACK! │
arp_observer  |  │ Detection Time     │ 00:38:55       │               │
arp_observer  |  └────────────────────┴────────────────┴───────────────┘
arp_observer  |
```

Figure 12: Output at observer PC

9 Security Assessment

9.1 Offensive Capabilities Analysis

9.1.1 Attack Effectiveness

1. **Network Disruption**: Successfully overwhelms target services
2. **ARP Cache Poisoning**: Effectively corrupts network device ARP tables
3. **Service Impact**: Causes victim services to become unresponsive
4. **Traffic Interception**: Enables MITM attacks through poisoning

9.1.2 Attack Sophistication

- Custom packet crafting without external attack tools
- Multi-threaded implementation for high packet rates
- Support for both flooding and targeted attacks
- Cross-platform compatibility (Linux/Windows)

9.2 Defensive Capabilities Analysis

9.2.1 Monitoring Effectiveness

1. **Comprehensive Coverage**: Monitors all ARP traffic patterns
2. **Real-time Alerting**: Immediate detection of suspicious activity
3. **Forensic Capabilities**: Detailed packet capture and analysis
4. **Scalable Architecture**: Handles high-volume traffic analysis

10 Ethical Considerations

10.1 Responsible Use Framework

1. **Legal Compliance**: Clear warnings about authorization requirements
2. **Controlled Environment**: Isolated testing within Docker containers
3. **Educational Purpose**: Explicit focus on learning and research
4. **Safety Measures**: Built-in confirmations and limited default parameters

10.2 Security Best Practices

- Only use on owned or explicitly authorized networks
- Follow responsible disclosure practices
- Document all testing activities
- Restore normal network state after testing

11 Technical Implementation Details

11.1 Codebase Structure

11.1.1 Attacker Implementation

- **arp_dos_storm.py**: Main Python attack engine with multi-threading
- **attacker_main.py**: Docker orchestration and interactive menu
- **utils.py**: Utility functions for network operations

11.1.2 Detection and Monitoring

- **arp_analyzer.py**: Core traffic analysis and detection engine
- **observer_main.py**: Container orchestration and live monitoring
- **monitor_main.py**: Web dashboard and visualization
- **victim_main.py**: Target service simulation

11.2 Performance Validation

11.2.1 Expected Outcomes

- Successful attack execution by Python
- Network disruption of victim services during attacks
- Accurate detection of attack patterns by observer
- Real-time monitoring data in web dashboard
- Achievement of expected packet generation rates
- Complete network isolation within Docker environment

12 Security Analysis and Risk Assessment

12.1 Vulnerability Assessment

12.1.1 ARP Protocol Vulnerabilities

1. **Lack of Authentication**: ARP packets are not authenticated
2. **Trust Model**: Devices automatically trust ARP responses
3. **Cache Overwriting**: ARP caches can be easily manipulated
4. **Broadcast Nature**: ARP operates on broadcast domain

12.2 Risk Mitigation Framework

12.2.1 Preventive Measures

1. **Network Segmentation**: Limit broadcast domain scope
2. **Static ARP Tables**: Use fixed mappings for critical systems
3. **ARP Monitoring**: Deploy continuous traffic analysis
4. **Rate Limiting**: Implement packet processing controls

12.2.2 Detective Controls

1. **Anomaly Detection**: Monitor for unusual ARP patterns
2. **Traffic Analysis**: Analyze packet characteristics
3. **Behavioral Monitoring**: Track device communication patterns
4. **Alert Systems**: Real-time notification of suspicious activity

13 Team Contributions

13.1 2005074 - Dipanta Kumar Roy Nobo

- Designed and implemented the Docker containerization architecture for the ARP attack simulation environment
- Configured the virtual network topology to enable realistic attack scenarios while maintaining isolation
- Developed the automated build system using Docker Compose for seamless deployment
- Implemented network namespace separation between attacker, victim, and observer containers
- Optimized container resource allocation to balance performance and system requirements

13.2 2005090 - Tawkir Aziz Rahman

- Architected the core ARP attack algorithms (storm and poisoning variants)
- Implemented comprehensive logging and monitoring systems across all containers
- Designed the statistical analysis framework for attack detection in the observer
- Developed the real-time visualization dashboard for attack monitoring
- Created the threshold-based detection mechanisms for different attack patterns

13.3 Collective Achievements

- Successfully demonstrated multiple ARP attack vectors in a controlled environment
- Developed defensive countermeasures for each attack type
- Created detailed documentation of attack methodologies and detection techniques
- Implemented a modular codebase allowing for future expansion of attack types
- Achieved measurable performance benchmarks for attack effectiveness

14 Conclusions and Recommendations

1. Successfully implemented comprehensive ARP attack simulation
2. Developed effective real-time detection and monitoring systems
3. Created isolated testing environment for safe experimentation
4. Provided complete educational framework for network security

15 Appendices

15.1 Appendix A: Installation and Setup

15.1.1 Prerequisites

- Docker Engine 20.10+
- Docker Compose 2.0+
- At least 2GB RAM
- Administrative/root privileges for raw sockets

15.1.2 Quick Start Commands

Listing 2: Setup Commands

```
# Build the lab environment
./lab-manager.sh build

# Start the lab
./lab-manager.sh start

# Access monitoring dashboard
# Browser: http://localhost:8080

# Run basic attack
./lab-manager.sh attack basic
```

15.2 Appendix B: Configuration Parameters

15.2.1 Attack Tool Parameters

Table 8: Attack Configuration Options

Parameter	Description	Default
–subnet	Target subnet	192.168.1
–duration	Attack duration (seconds)	60
–threads	Number of threads	4
–rate	Packets per second per thread	100
–interface	Network interface	eth0

15.2.2 Detection Thresholds

Table 9: Configurable Detection Parameters

Threshold	Default Value	Description
packets_per_second	50	ARP packets per second limit
unique_senders_per_minute	20	Unique MAC addresses per minute
gratuitous_ratio	0.7	Percentage of gratuitous ARP
mac_changes_per_ip	3	MAC changes per IP in 5 minutes

15.3 Appendix C: Legal and Ethical Guidelines

15.3.1 Legal Considerations

- Only use on networks you own or have explicit written permission
- Unauthorized network attacks are illegal in most jurisdictions
- This tool is for educational and authorized testing purposes only
- Always follow responsible disclosure practices

15.3.2 Ethical Guidelines

- Use isolated testing environments
- Start with minimal parameters
- Monitor impact during testing
- Document all testing activities
- Restore normal network state after testing

16 References

1. RFC 826: Ethernet Address Resolution Protocol
2. NIST Cybersecurity Framework
3. OWASP Network Security Testing Guide
4. IEEE 802.3 Ethernet Standards
5. Docker Security Best Practices
6. Network Security Assessment Methodologies