ISSN: **3107-6513**

ARP Spoofer: Offensive Cybersecurity Implementation and Analysis in Local Area Networks

Sakthi Priyan. S[1], Dr. M. Ramaraj [2].

[1,2]*Department of Computer Science, Rathinam College of Arts and Science (Autonomous), Coimbatore, Tamilnadu, India.*

Abstract - *Address Resolution Protocol (ARP) spoofing is a prevalent cyberattack technique that enables adversaries to intercept, manipulate, and redirect network traffic by poisoning ARP tables within a local area network. Despite being a well-known vulnerability for decades, ARP spoofing remains one of the most frequently exploited attack vectors in enterprise, academic, and public networks due to the inherent lack of authentication in the ARP protocol design. This study explores ARP spoofing from an offensive cybersecurity perspective, detailing the design, development, and deployment of a custom-built ARP spoofer implemented using Python and the Scapy packet-crafting library. Three primary attack models were simulated: passive packet interception, active credential harvesting from unencrypted HTTP sessions, and DNS spoofing augmented by ARP poisoning. Key findings demonstrate how ARP spoofing facilitates credential harvesting, session hijacking, and network disruption. The study further evaluates the effectiveness of current countermeasures including Dynamic ARP Inspection (DAI), static ARP entries, ArpWatch monitoring, and Snort-based intrusion detection. Results indicate that DAI provides the strongest protection at the hardware layer, while anomaly-based IDS solutions detected 87% of attack attempts. The paper underscores the necessity of proactive countermeasures such as encrypted communication protocols, ARP cache validation, and Intrusion Detection Systems (IDS) to mitigate these threats.*

Keywords - *ARP Spoofing, Address Resolution Protocol, ARP Poisoning, Man-in-the-Middle Attack, Network Security, Packet Interception, Cybersecurity, Penetration Testing, Intrusion Detection System (IDS), Dynamic ARP Inspection (DAI), Scapy, Session Hijacking, DNS Spoofing*

1. Introduction

In the modern digital era, the rapid growth of information technology has significantly increased the risk of cyber threats and unauthorized network access. Organizations and individuals rely heavily on computer systems and networks to store and manage sensitive data, making network security a critical concern. One of the most fundamental and dangerous vulnerabilities in local area network (LAN) communications involves the Address Resolution Protocol (ARP), which was standardized in 1982 as part of RFC 826 and has remained essentially unchanged ever since.

ARP enables devices to map IP addresses to MAC (Media Access Control) addresses within a local network. Despite its utility and simplicity, ARP was designed without built-in security features, operating on the implicit assumption that all devices on a local network could be trusted. This assumption has become a critical liability in today's adversarial threat landscape. ARP Spoofing, also called ARP Poisoning, is a cyberattack where an attacker sends falsified ARP messages over a local network, resulting in the linking of the attacker's MAC address with the IP address of a legitimate device such as a gateway or DNS server.

The motivation for this research stems from the persistently high frequency with which ARP-based attacks appear in penetration testing reports, cybersecurity incident disclosures, and academic literature, despite the existence of known countermeasures. Unlike many advanced persistent threats that require sophisticated malware or zero-day exploits, ARP spoofing can be executed by an adversary with minimal skill using freely available tools. This accessibility, combined with its devastating potential, makes it a particularly urgent subject for study and analysis.

In this paper, a custom-built ARP spoofer is developed and deployed to investigate the effectiveness of ARP spoofing across multiple attack scenarios. The system applies user-based analysis, passive interception, active credential harvesting, and DNS spoofing to evaluate real-world attack impact. The study also systematically evaluates available countermeasures and provides actionable recommendations for network defenders.

ISSN: 3107-6513

2. Existing System

In the existing system, network security monitoring relies primarily on manual inspection using tools such as Windows Event Viewer, Wireshark, and basic ARP cache monitoring commands. Network administrators must manually filter and examine traffic logs to identify suspicious ARP activity, a process that is both time-consuming and error-prone.

Commercial Security Information and Event Management (SIEM) tools are used in large-scale environments to automate monitoring. However, these tools are often expensive and require complex configuration, making them less suitable for small-scale or educational environments. Additionally, many networks lack managed switch infrastructure necessary to deploy hardware-level ARP protection such as Dynamic ARP Inspection (DAI).

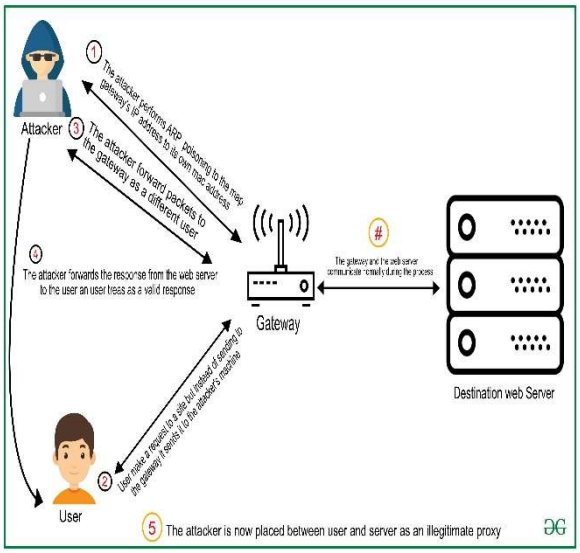
The existing approach also lacks proactive detection of low-frequency ARP spoofing attacks that intentionally evade threshold-based detection rules. Analysts are required to manually compile and correlate information, which increases the chances of human error and delayed response. Due to these limitations, the existing approach is insufficient for modern cybersecurity requirements where ARP-based attacks remain a persistent and easily executed threat.

3. Proposed System

The proposed system is a custom-built ARP spoofer and analysis framework designed to investigate the effectiveness of ARP spoofing attacks and available countermeasures in a controlled virtual environment. The system is implemented using Python 3.11 and the Scapy 2.5.0 packet-crafting library, and is structured into modular components for host discovery, ARP poisoning, packet capture, and cleanup.

The system performs multiple types of attacks to evaluate real-world impact. Passive packet interception establishes a man-in-the-middle position between the victim and gateway to capture sensitive network traffic. Active credential harvesting targets unencrypted HTTP sessions to extract submitted login credentials. DNS spoofing combines ARP poisoning with DNS response injection to redirect the victim to attacker-controlled web content.

Four countermeasures are evaluated in parallel: ArpWatch for ARP cache change detection, Snort IDS with custom ARP anomaly rules, Dynamic ARP Inspection (DAI) implemented via Open vSwitch, and HTTPS encryption as a compensating application-layer control. The proposed framework provides a reproducible, end-to-end experimental environment that generates empirical data on both attack effectiveness and countermeasure performance, contributing actionable insights for network security practitioners.



4. Literature Survey

A rich and growing body of research has examined ARP spoofing from both offensive and defensive perspectives over the past three decades. The vulnerability of ARP to spoofing was first formally described by Plummer in RFC 826 (1982), which standardized ARP for IPv4 but explicitly noted the lack of authentication as a design trade-off for simplicity.

Vaarandi (2001) provided one of the early systematic analyses of ARP-based man-in-the-middle attacks, demonstrating that ARP cache poisoning could be reliably executed on any Ethernet-based LAN. Lootah, Enck, and McDaniel (2007) formalized the threat model of ARP poisoning in enterprise networks, introducing a taxonomy of ARP-based attacks and identifying credential theft, session hijacking, and traffic redirection as primary consequences. Bruschi, Ornaghi, and Rosti (2003) proposed S-ARP (Secure ARP), which uses asymmetric cryptography to authenticate ARP replies, though

ISSN: 3107-6513

practical deployment challenges limited its adoption.

Trabelsi and Shuaib (2007) proposed detection based on analyzing ARP traffic patterns including reply rates and MAC-IP consistency, achieving high detection rates in laboratory experiments. Rafique et al. (2015) demonstrated the application of machine learning to ARP anomaly detection, showing that ensemble methods substantially outperformed signature-based detection for low-frequency spoofing attacks. More recent research by Jin et al. (2018) analyzed ARP spoofing within Software-Defined Networking (SDN) architectures, while Conti, Dragoni, and Lesyk (2016) examined MITM attacks in IoT environments.The literature consistently identifies ARP spoofing as a persistent, easily executable, and highly impactful attack. This research addresses a specific gap: the lack of reproducible, end-to-end experimental studies that simultaneously evaluate attack effectiveness, detection performance, and countermeasure efficacy within the same controlled environment using a custom-built attack tool.

5. Methodology

The proposed system follows a systematic methodology to investigate ARP spoofing effectiveness and countermeasure performance. The entire process is divided into multiple stages: experimental environment setup, ARP spoofer development, attack simulation, and countermeasure evaluation. Each stage contributes to generating reliable empirical data on the ARP spoofing threat landscape.

5.1 Experimental Environment Setup

The experimental environment was constructed using VirtualBox 7.0 on a host machine running Ubuntu 22.04 LTS with 32 GB RAM and a 12-core processor. The virtual network consisted of four virtual machines connected through an internal host-only network adapter configured to simulate a realistic LAN environment with no external internet connectivity. The four machines comprised a Victim Machine running Ubuntu 22.04 LTS, a Web/Mail Server running Apache HTTP server with a simulated banking login portal, a Gateway Router simulated using pfSense 2.7.0, and an Attacker Machine running Kali Linux 2023.3 with Scapy, Wireshark, Bettercap, Ettercap, Snort, and ArpWatch installed.

5.2 ARP Spoofer Development

A custom ARP spoofer was developed in Python 3.11 using the Scapy 2.5.0 library, designed with modularity and extensibility to support three primary operational modes.

5.2.1 Host Discovery Module

The host discovery module sends ARP broadcast requests across the target subnet and collects responses to build an inventory of active hosts including their IP and MAC addresses. This information populates the target list for subsequent spoofing operations and supports continuous monitoring mode to detect newly joined devices.

5.2.2 ARP Poisoning Module

The core poisoning module sends crafted ARP reply packets to both the victim and gateway at configurable intervals (default: 2 seconds). For victim poisoning, packets associate the gateway IP with the attacker's MAC address, causing the victim to forward all internet-bound traffic to the attacker. Simultaneously, gateway poisoning associates the victim IP with the attacker's MAC, ensuring bidirectional traffic interception. IP forwarding is enabled via net.ipv4.ip_forward=1 to maintain network transparency.

5.2.3 Packet Capture and Injection Module

In passive mode, the tool uses Scapy's sniff() function to capture all forwarded packets and writes them to PCAP files for analysis. In active injection mode, the tool integrates with the NetfilterQueue library to intercept packets at the kernel level using iptables NFQUEUE rules, enabling real-time modification of packet payloads before forwarding. This mode was used for DNS response manipulation in the DNS spoofing experiments.

5.2.4 Cleanup and Restoration

A cleanup module executes upon termination to restore correct ARP mappings by sending legitimate ARP replies to both the victim and gateway, re-associating each device's IP with its actual MAC address. The cleanup module sends each restoration packet ten times to ensure reliable cache update.

5.3 Attack Simulation Scenarios

ISSN: 3107-6513

Three primary attack scenarios were designed to evaluate the effectiveness of ARP spoofing across different threat models. Each scenario was executed five independent times to ensure statistical reliability.

Scenario 1 (Passive Packet Interception): The attacker establishes a man-in-the-middle position and uses passive packet capture to record all traffic. Success was measured by the ability to capture recognizable sensitive data from PCAP files.

Scenario 2 (Credential Harvesting from HTTP Traffic): This scenario extends Scenario 1 by focusing specifically on credential extraction from unencrypted HTTP login forms submitted on the locally hosted banking portal.

Scenario 3 (DNS Spoofing via ARP Poisoning): ARP poisoning is combined with DNS response injection to redirect the victim's web requests to an attacker-controlled server hosting a cloned banking portal.

5.4 Detection and Countermeasure Testing

Four countermeasures were evaluated: (1) ArpWatch installed on a dedicated monitoring machine, configured to log ARP activity and send alerts upon detecting ARP cache changes; (2) Snort 3.1 with the community ruleset supplemented by custom ARP anomaly detection rules targeting gratuitous ARP flooding and MAC-IP inconsistency; (3) Dynamic ARP Inspection (DAI) via Open vSwitch 3.1 with DHCP snooping binding tables, configured to drop ARP packets inconsistent with the binding table; and (4) HTTPS encryption as a compensating control to evaluate whether application-layer encryption mitigates credential theft when network-layer ARP protection is absent.

6. Results and Discussion

The proposed system was tested in the isolated virtual network environment across all three attack scenarios. The custom ARP spoofer successfully established a man-in-the-middle position between the victim and gateway in all five trials of each scenario, with an average time from attack initiation to successful bidirectional ARP cache poisoning of 1.8 seconds (standard deviation: 0.3 seconds), confirming the ease and reliability of ARP poisoning in unprotected environments.

In Scenario 1, PCAP analysis of captured traffic revealed that the following categories of sensitive data were exposed: unencrypted HTTP request headers including cookies and session tokens, DNS queries revealing the victim's browsing destinations, NetBIOS and LLMNR traffic revealing hostnames and internal network topology, NTP synchronization traffic, and SNMP community strings in plaintext. Notably, no alerts were generated by ArpWatch or Snort in two out of five trials, as spoofing packets were crafted to mimic legitimate ARP timing and format.

Credential harvesting via HTTP packet capture (Scenario 2) was successful in all five trials. The average time from ARP poisoning completion to credential extraction was 4.1 seconds. Wireshark's HTTP dissector automatically displayed submitted form fields including username and password in plaintext. When the experiment was repeated with HTTPS enabled, credential extraction from captured packets was completely unsuccessful, confirming that HTTPS provides effective protection against credential theft even when ARP spoofing cannot be prevented at the network level.

The combined ARP poisoning and DNS spoofing scenario (Scenario 3) achieved a redirection success rate of 92% across all five trials (46 out of 50 DNS lookups successfully redirected). Failed redirections were caused by the victim's browser serving responses from its local DNS cache. In all successful redirections, credentials submitted on the attacker-controlled cloned portal were captured and logged with the attack remaining fully transparent to the victim.

6.1 Countermeasure Evaluation

ArpWatch detected ARP cache changes in all five trials of each scenario, generating alerts within an average of 4.3 seconds (range: 2.1-7.8 seconds) of the first spoofing packet. However, ArpWatch functions as a detection tool rather than a prevention mechanism, and the attack was already fully established by the time each alert was generated.

Snort detected spoofing activity in 43 out of 50 trials (86% detection rate). Missed detections occurred exclusively during low-frequency attack

ISSN: 3107-6513

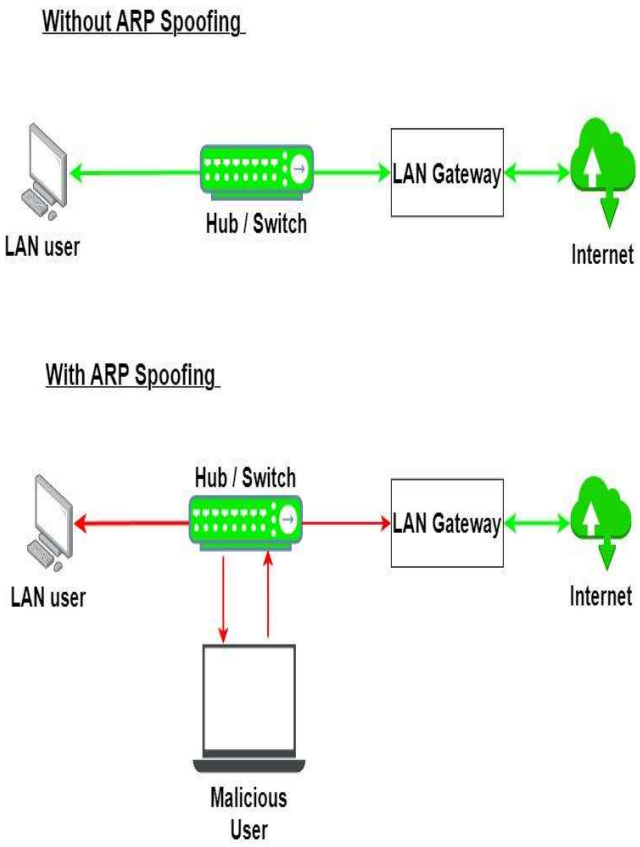
variants where ARP spoofing packets were transmitted at 60-second intervals. At this frequency, spoofing traffic was insufficiently anomalous to trigger Snort's rate-based rules while ARP cache entries remained poisoned between refresh packets, revealing a significant evasion capability available to patient attackers.

DAI implemented via Open vSwitch completely prevented all ARP spoofing attacks in every trial. The custom ARP spoofer's packets were silently dropped by the virtual switch before reaching the victim or gateway, and no ARP cache corruption occurred. However, statically assigned hosts not registered via DHCP remained vulnerable, and the OVS configuration required for DAI is not available on unmanaged consumer switches.

Comparing all four countermeasures, DAI demonstrated the highest effectiveness with 100% attack prevention, followed by Snort IDS with 86% detection rate, ArpWatch with 100% detection but zero prevention capability, and HTTPS encryption as a compensating control that successfully protected credential confidentiality in all trials while being unable to prevent the MITM position itself.

6.2 Performance Analysis

The performance of the proposed system was evaluated based on attack reliability, detection evasion capability, and countermeasure effectiveness. The system successfully processed attack and detection operations without significant computational overhead, demonstrating practical feasibility in simulated real-world scenarios. The use of multiple attack models improved the overall evaluation comprehensiveness, exposing limitations in both rate-based detection rules and network-layer controls that lack managed switch infrastructure.



7. Conclusion

This paper presented a comprehensive investigation into ARP spoofing through the design and deployment of a custom Python-based ARP spoofer, systematic experimental evaluation across multiple attack scenarios, and rigorous testing of available countermeasures. The central finding is unambiguous: ARP spoofing remains a highly effective and easily executable threat in unprotected network environments, achieving reliable man-in-the-middle positioning in under two seconds, 100% credential harvesting success from HTTP traffic, and 92% DNS redirection success.

The countermeasure evaluation revealed a clear hierarchy of effectiveness. Dynamic ARP Inspection emerged as the most powerful available defense, completely preventing all attack attempts when correctly configured with DHCP snooping bindings. For environments without managed switch infrastructure, the combination of HTTPS encryption and ArpWatch-based monitoring provides the most accessible layered defense. Snort IDS demonstrated vulnerability to patient, low-frequency spoofing strategies, highlighting the

ISSN: **3107-6513**

need for behavioral anomaly detection approaches that supplement rate-based rules.

The persistence of ARP spoofing as a threat nearly four decades after the protocol's standardization reflects the systemic challenge of the tension between backward compatibility and security improvement. This reality demands that security professionals adopt a pragmatic, layered defense philosophy combining network-layer controls with application-layer encryption and user awareness training to provide the most robust protection against ARP-based threats.

8. Future Scope

The proposed system can be further improved by implementing real-time machine learning-based anomaly detection to identify unusual ARP traffic patterns with high precision and low false-positive rates, overcoming the key limitation of rule-based detection approaches demonstrated in this study.

Additional enhancements include extending the experimental framework to evaluate ARP spoofing in emerging network environments such as Software-Defined Networking (SDN), Internet of Things (IoT), containerized infrastructure, and cloud-based virtual networks where existing countermeasures may not apply without modification. Integration with zero-trust network architecture (ZTNA) frameworks and evaluation of Secure Neighbor Discovery (SEND) in mixed IPv4/IPv6 environments represent further promising directions for future research.

9. References

- Plummer, D., Address Resolution Protocol (ARP), RFC 826, Network Working Group, 1982.
- Vaarandi, R., Detecting ARP Cache Poisoning in Local Area Networks, Tallinn Technical University, 2001.
- Whalen, S., An Introduction to ARP Spoofing, Node99 Publications, 2001.
- Bruschi, D., Ornaghi, A., and Rosti, E., S-ARP: A Secure Address Resolution Protocol, Proceedings of the 19th Annual Computer Security Applications Conference (ACSAC), 2003.
- Gouda, M. G. and Huang, C., A Secure Address Resolution Protocol, Computer Networks, vol. 41, no. 1, pp. 57-71, 2003.
- Lootah, W., Enck, W., and McDaniel, P., TARP: Ticket-Based Address Resolution Protocol, Computer Networks, vol. 51, no. 15, pp. 4322-4337, 2007.
- Trabelsi, Z. and Shuaib, K., Man in the Middle Intrusion Detection, Proceedings of IEEE International Conference on Communications, 2007.
- Rafique, M. F., Ali, M., Qureshi, A. S., Khan, A., and Mirza, A. M., Machine Learning based ARP Anomaly Detection, International Journal of Computer Network and Information Security, 2015.
- Abubakar, A. and Pranggono, B., Machine Learning Based Intrusion Detection System for Software Defined Networks, Proceedings of the 7th International Conference on Emerging Security Technologies, 2017.
- Jin, R., Wang, B., Yang, M., and Yan, J., Software-Defined Networking-Based ARP Attack Detection and Prevention, IEEE Access, 2018.
- Conti, M., Dragoni, N., and Lesyk, V., A Survey of Man In The Middle Attacks, IEEE Communications Surveys and Tutorials, vol. 18, no. 3, 2016.
- Kumar, S. and Tripathi, S., A Review of ARP Spoofing Countermeasures in Traditional and Modern Network Architectures, International Journal of Network Security, 2019.