Security Enhancement for MITM and DDoS Attacks in IoT Networks

Joel Kannan 1, a), Sumendra Yogarayan 1, 2, b), Ang Ee Mae 1, 2, c), and Thinesh Ganesan 1, 3, d)

1 Faculty of Information Science and Technology, Multimedia University, Melaka 75450 Malaysia

2 Centre for Intelligent Cloud Computing, COE for Advanced Cloud, Multimedia University, Melaka 75450 Malaysia

3 Centre for Advanced Analytic, COE for Artificial Intelligence, Multimedia University, Melaka 75450 Malaysia

*b) Corresponding author: sumendra@mmu.edu.my*

*a) joeljr4827@gmail.com*

*c) eemae@mmu.edu.my*

*d) thinesh.ganesan@mmu.edu.my*

**Abstract.** The increasing use of Internet of Things (IoT) devices has expanded connectivity in the digital ecosystem. However, this growth has also exposed existing security vulnerabilities, especially the system’s vulnerability to man-in-the-middle (MITM) attacks and distributed denial-of-service (DDoS) attacks. This study aims to improve the security of IoT networks by implementing specific defense mechanisms for these threats. Analysis of an existing IoT prototype reveals vulnerabilities to Address Resolution Protocol (ARP) poisoning (a type of MITM attack) and SYN TCP flood attacks (a type of DDoS attack). To address these vulnerabilities, we implement the ARP traffic monitoring utility to prevent ARP poisoning and configure an IP schedule firewall to mitigate SYN TCP flood attacks, thereby strengthening network security. Penetration tests provide an in-depth look at the vulnerabilities of the IoT prototype. Finally, we present comparative results that show how these mitigations successfully reduce half-open TCP connections and CPU usage spikes during attacks.

# Introduction

Internet of Things (IoT) is a transformative paradigm that connects physical objects to the internet to enable seamless data transfer. From household appliances to wearables and industrial machines, these connected devices enable intelligent decision-making and automation in areas as diverse as healthcare, transportation, manufacturing, and smart homes [1]. IoT devices leverage communication protocols, sensors, and software to collect, process, and transmit data to a central system, often with the help of artificial intelligence to improve performance. While IoT holds great potential for increased convenience and connectivity, this paradigm also poses major challenges, particularly in ensuring interoperability, scalability, and security [2].

The IoT devices are resource-constrained (computing and memory), which increases their susceptibility to cyber threats. MITM attacks that target communication protocol vulnerabilities in the IoT ecosystem enable an adversary to block, alter, or otherwise tamper with the communication between IoT devices and their targets [3]. Instead, DDoS attacks generate a bulk of illegitimate network traffic through botnets that overwhelm the target devices and render the system unusable resulting in service loss [4]. These threats run the risk of breaching sensitive information, an important system breakdown, and loss of consumer confidence. The security technologies related to the IoT have progressed quite fast, but substantial vulnerabilities are intact, mainly because the majority of the IoT devices do not have in-built encryption features.

# Literature Review

IoT has revolutionized industries because it has enabled them to connect freely with other physical devices, software, and sensors [6]. This interconnected network enables real-time data sharing and automation, which is transforming across fields such as healthcare, smart home, transportation, and manufacturing. Smith et al. [7] and Chen et al. [8] highlight the increasing popularity of DDoS attacks on IoT devices that are frequently caused by inadequate security settings. The amount of traffic caused during DDoS attacks causes problems where the systems cannot distinguish between genuine and malicious requests causing disruption of services. To control such threats, researchers have suggested the use of the advanced traffic monitoring and filtering systems where it can distinguish itself among the malicious traffic in real time using the Software-Defined Networking (SDN) middleware [9].

Lee and Gupta [10] point out, most IoT systems have improperly implemented security measures in the form of default passwords, lack of encryption (e.g. Transport Layer Security (TLS) or Elliptic Curve Cryptography (ECC)), making them vulnerable to both MITM and DDoS attacks. The integrity of communication channels between devices is one of the issues of the MITM attacks alleviation since an attacker can alter the information being transferred or inject malicious instructions. Enhancing encryption and implementing secure communication standards (e.g. TLS/ECC) are valuable measures in the mitigation of MITM attacks [11].

Patel and Li [12] and Fernandez and Murphy [13] have deliberated the application of machine learning (ML) models, including artificial neural network (ANN) and decision tree (DT) to detect anomalies in the IoT network traffic, allowing earlier detection of potential DDoS or MITM attacks. These systems are able to monitor traffic patterns and alert anomalies that represent deviation from normal behaviour and thus cause automated responses thereby containing an attack before it gets worse. Furthermore, a study conducted by Zhang et al. [14] confirms the assertion that strong encryption is important when ensuring data integrity. Such protocols as ECC and TLS have the ability to deny unauthorized access and manipulation with sensitive data. The use of two-factor authentication (2FA) as regular updates of firmware or software are also cited as good practices in IoT ecosystem security [15].

In addition to software-driven measures, hardware-based protections such as firewalls and intrusion detection systems (IDS) are important components in safeguarding IoT networks. Firewalls can block malicious traffic based on predefined security rules, while IDS monitor network traffic for suspicious activity in real time [16]. Recent studies also highlights the potential of decentralized security solutions such as blockchain to protect IoT devices. Blockchain distributed ledger system offers a tamper-proof method for storing and sharing data, ensuring data authenticity and transparency, especially in supply chain applications [17].

While these advanced techniques (ML, SDN, ECC, blockchain) show promise, the focus of this project is on practical, fundamental countermeasures that can be deployed on resource-constrained IoT devices. By leveraging lightweight tools such as ARPWatch for ARP poisoning detection and iptables for TCP SYN flood mitigation, we demonstrate simple configured measures that can substantially improve IoT security.

## PROPOSED WORK

In this project, we design and implement security measures to protect IoT devices against common attacks, specifically focusing on Man-in-the-Middle (MITM) attacks via ARP poisoning and Distributed Denial of Service (DDoS) attacks via TCP SYN flooding. Because many IoT devices have limited resources (e.g., CPU and memory), both preventive and responsive measures must be lightweight yet effective.

ARP poisoning is a form of MITM in which an attacker sends forged ARP messages on a local network, associating their own Media Access Control (MAC) address with the IP address of a legitimate device (router or gateway). This enables the attacker to intercept, modify, or redirect packets between two communicating devices. In an IoT context, devices such as smart thermostats, cameras, and sensors share a local network where successful ARP poisoning can compromise sensitive data (e.g., user credentials, sensor readings) or disrupt operations. To mitigate ARP poisoning, we deploy ARPWatch, a utility that monitors ARP traffic in real time, logs changes in IP-to-MAC mappings, and raises alerts when anomalous ARP events occur. With ARPWatch installed on a Linux host, administrators receive immediate notifications of suspicious ARP entries, allowing them to block or investigate potential MITM attempts

A TCP SYN flood is a type of DDoS attack that targets the TCP handshake. The attacker sends a large volume of TCP synchronize (SYN) packets with spoofed source IP addresses to the victim. Each SYN packet causes the target to allocate memory and respond with a SYN-ACK packet, then await the final ACK from the (non-existent) client. Because the attacker never completes the handshake, the victim’s backlog of half-open TCP connections grows until resources (e.g., memory, CPU) are exhausted, causing service disruption. To counteract SYN floods, we configure iptables on a Kali Linux host to rate-limit incoming SYN packets and drop excess SYN-ACK responses. For example, an iptables rule might allow one new SYN per second (burst of three) and drop further SYN packets. With the correct rate limiting and packet filtering rules, the victim’s backlog is kept minimal, preventing resource exhaustion.

Figure 1 outlines the flow of data transmission, threat detection, analysis, and prevention. It demonstrates how the system will continuously check for potential threats, and based on the analysis, will either prevent real threats or ignore false threats.

A diagram of a data security system

Description automatically generated

**Figure 1.** Process flow of threat detection and prevention in an IoT system

## IMPLEMENTATION

## Man-in-the-Middle (MITM) Attacks

The first attack is ARP poisoning, a MITM technique. In ARP poisoning, an attacker sends forged ARP messages on a local network to associate the attacker’s MAC address with the IP address of another device (e.g., the router). Any traffic intended for that IP is then delivered to the attacker. Because IoT devices (e.g., smart cameras, thermostats, or healthcare monitors) often share the same local network, they typically trust ARP responses without verification.

**Victim Setup:** We simulated a victim IoT device on Raspbian OS (running on a Raspberry Pi). The IoT device communicated with a standard router (gateway) over an Ethernet or Wi-Fi network segment.

**Attack Execution:** The attacker used **Ettercap** on a Kali Linux machine to launch ARP poisoning. Ettercap sent malicious ARP packets to both the victim and the router, poisoning their ARP caches:

* The victim’s ARP cache associated the attacker’s MAC with the router’s IP.
* The router’s ARP cache associated the attacker’s MAC with the victim’s IP.  
  As a result, all traffic between the victim device and router flowed through the attacker’s machine.

**Traffic Capture:** With the ARP poisoning in place, the attacker launched **Wireshark** on the Kali Linux host to capture traffic. Figure 2 shows Wireshark capturing packets purportedly from the victim. In this IoT scenario, sensitive data (e.g., login credentials, sensor readings) can be intercepted.

A screenshot of a computer

Description automatically generated

**Figure 2.** Wireshark monitoring after ARP poisoning

**Demonstration of Compromise:** Figure 3 illustrates how Wireshark reveals user login credentials or other IoT data once MITM is successful. In an actual deployment (e.g., a smart home), the attacker could alter thermostat settings or access security camera feeds.

A screenshot of a computer

Description automatically generated

**Figure 3.** Process of attacker captured victim activity through WireShark

**ARPWatch Deployment:** To prevent ARP poisoning, **ARPWatch** was installed on a dedicated monitoring host (also running Kali Linux). ARPWatch passively listens to ARP packets and maintains a database of IP-to-MAC mappings. When a new MAC is seen for a known IP (or vice versa), ARPWatch logs the event and can be configured to send an email or console alert. Figure 4 shows the selection from the journalctl logs, where ARPWatch detected a MAC change, indicating a possible ARP poisoning attempt.

A screen shot of a computer

Description automatically generated

**Figure 4.** MAC address changes monitoring and alert trigger

When ARPWatch raises an alert, the network administrator can take immediate action such as blocking the attacker’s MAC at the switch or using a host-based firewall rule. In our testing, ARPWatch reliably detected all forged ARP entries during the simulated poisoning attempts, with zero false negatives. Because ARPWatch requires minimal CPU and memory, it is well suited for small-scale IoT monitoring hosts.

## Distributed Denial of Service (DDoS) Attacks

The second attack is a TCP SYN flood, a common form of DDoS. In a TCP SYN flood, the attacker sends a large number of TCP SYN packets to the victim’s open ports with spoofed source IP addresses. Each SYN packet forces the victim to allocate resources for a half-open connection *(state =SYN\_RECEIVED)*, waiting for an ACK that never arrives. As the victim’s backlog of half-open connections grows, CPU usage and memory are exhausted, ultimately causing a denial of service.

**Attack Execution with HPING:** The attacker used **HPING** on Kali Linux to craft a flood of TCP SYN packets. The command included the *--rand-source* option, which generates random source IP addresses for each packet, amplifying the attack’s stealth and making filtering harder. In our lab setup, the target IoT device’s IP was *192.168.220.129*. Figure 5 shows HPING sending thousands of SYN packets per second to the victim.

A screenshot of a computer

Description automatically generated

**Figure 5.** Process of SYN Flood Attack

**Traffic Monitoring and Impact:** With Wireshark capturing on the victim, we observed a flood of TCP SYN packets arriving. Each triggered a SYN-ACK response from the victim, but because the attacker never sent the final ACK, those connections remained half-open. Figure 6 shows Wireshark’s view of the victim’s SYN-ACK replies. Over time, the number of half-open connections grew exponentially, and CPU usage (monitored via top) spiked from ~5 % utilization to >90 %, causing network service slowdowns.

A screenshot of a computer

Description automatically generated

**Figure 6.** Wireshark monitoring after SYN-ACK flooding packets

**iptables Mitigation:** To mitigate the SYN flood, we configured **iptables** on the victim host:

|  |
| --- |
| sudo iptables -A INPUT -p tcp --syn -m limit --limit 1/s --limit-burst 3 -j ACCEPT  sudo iptables -A INPUT -p tcp --syn -j DROP |

This rule allows up to one new SYN packet per second, with a burst allowance of three. Any further SYN packets are dropped.

**Blocking SYN-ACK responses:**

|  |
| --- |
| sudo iptables -A OUTPUT -p tcp --tcp-flags SYN,ACK SYN,ACK -j DROP |

By blocking SYN-ACK packets from leaving the victim, we prevent an attacker from forcing the victim into half-open TCP states.

**Post-mitigation Verification:** After applying the iptables rules, we repeated the HPING flood. Using *netstat -tn | grep SYN\_RECV | wc -l*, we counted half-open connections before and after mitigation. Before mitigation, the count quickly rose above 500 in seconds; after mitigation, it remained below 10. Likewise, CPU usage stabilized around 15% instead of spiking above 90%. This demonstrates that simple iptables rate limiting and packet filtering can effectively neutralize a SYN flood on resource-constrained devices.

# Result and analysis

In this section, we provide observations from our experiments. We compare system behavior before and after implementing ARPWatch (for MITM protection) and iptables rules (for DDoS protection). All tests were performed on identical hardware: a Raspberry Pi 4 running Raspbian OS and Kali Linux hosts on the same network segment.

## ARPWatch (MITM Mitigation)

During our ARP poisoning simulation, where forged ARP packets were injected at a rate of 5 packets per second over a two-minute interval. The ARPWatch running on the monitoring host generated exactly twenty alerts, corresponding to each spoofed ARP update. In every run of the experiment, there were no missed detections: ARPWatch reliably flagged every malicious IP-to-MAC mapping change, indicating a detection accuracy of 100 %. In terms of resource overhead, the ARPWatch daemon consistently consumed less than 1 % of a Raspberry Pi’s CPU and under 50 MB of RAM, making it invisible to other processes. The average latency between the first forged ARP packet and the appearance of an alert on the administrator’s console was under half a second, allowing real-time intervention. Overall, these results show that ARPWatch can detect ARP poisoning attempts immediately and with slight computational cost, making it a practical choice for lightweight IoT network monitoring.

## iptables (SYN Flood Mitigation)

When exposing the victim host to a TCP SYN flood of 10,000 packets per second, we observed that, without mitigation, the number of half-open TCP connections (in the SYN\_RECV state) rose quickly, peaking at approximately 520 concurrent half-open connections within a few seconds. Correspondingly, CPU utilization on the Raspberry Pi spiked from a nominal 5% idle state to over 90%, causing service degradation and nearly to total unresponsiveness. After applying our iptables rules (one SYN per second with a burst of three allowed, and dropping excess SYNs and outgoing SYN-ACKs), the peak number of half-open connections remained below 10 throughout the same attack duration. Under this mitigation, CPU usage stabilized around 15%, and the device remained responsive. In effect, iptables dropped more than 99% of malicious SYN packets, reducing half-open connection buildup by roughly 98% and keeping resource consumption within acceptable limits for a resource-constrained IoT device.

## Limitation

Although ARPWatch and iptables proved effective in our laboratory environment, there are several limitations inherent to this work. First, our focus was limited to ARP poisoning and TCP SYN flood attacks; other attack vectors such as DNS spoofing, jamming at the physical layer, or more sophisticated application-layer exploits were not addressed here. Second, by design, ARPWatch and iptables represent baseline, they do not incorporate adaptive or machine-learning-driven anomaly detection. As a result, an attacker who can bypass ARPWatch alerts, for example, by flooding ARP traffic faster than the monitoring host can process or who employs multi-vector DDoS techniques may still succeed in causing disruption. Additionally, our experiments were conducted in a controlled lab environment using a single Raspberry Pi and a limited network topology; real-world IoT deployments may involve hundreds or thousands of devices distributed across multiple subnets, where latency, switch-level filtering, and differing hardware capabilities could affect detection times and mitigation efficacy. Within these constraints, our study remains scoped to demonstrate how lightweight, readily available tools can reduce the impact of two very common network attacks in small-scale IoT settings.

# CONCLUSION

This study focused on implementing security measures to protect IoT devices from common network attacks specifically Man-in-the-Middle (MITM) via ARP poisoning and Distributed Denial of Service (DDoS) via TCP SYN flooding. We demonstrated the practical use of ARPWatch to detect and prevent ARP poisoning, ensuring network communications remain secure. In addition, iptables were configured to mitigate TCP SYN flooding attacks, effectively reducing half-open TCP states and preserving system stability. Future work will include a wide range of attacks targeting IoT devices, such as DNS spoofing, physical layer attacks, and malware infections. Comprehensive threat modeling and mitigation techniques for these attacks will provide a robust security framework for the IoT ecosystem. The work could also examine the integration of machine learning algorithms to enhance detection and prevention of attacks. AI-based models can help identify attack patterns in IoT networks, provide adaptive and proactive security measures.

# Acknowledgments

I would like to express my gratitude to the reviewers and all those who indirectly supported this project. Their constructive feedback and the resources available made the successful completion of this work possible.

# References

1. H. Tran-Dang and D. S. Kim, “The physical internet in the era of digital transformation: perspectives and open issues,” *IEEE Access* **9**, 164–613 (2021).
2. S. Singh, G. Madaan, A. Singh, D. Pandey, A. S. George, and B. K. Pandey, “Empowering connectivity: exploring the internet of things,” in *Interdisciplinary Approaches to AI, Internet of Everything, and Machine Learning*, (IGI Global Scientific Publishing, 2025), pp. 89–116.
3. J. Cecílio and A. Souto, “Security issues in industrial Internet-of-Things: Threats, attacks and solutions,” in *Proc. IEEE Int. Workshop Metrol. Ind. 4.0 & IoT (MetroInd 4.0 & IoT)* (IEEE, 2024), pp. 458–463.
4. V. Tyagi, A. Saraswat, A. Kumar, and S. Gambhir, “Securing IoT devices against MITM and DoS attacks: An analysis,” in *Reshaping Intelligent Business and Industry: Convergence of AI and IoT at the Cutting Edge* (2024), pp. 237–249.
5. S. Shivaji, “DDoS attack detection: Strategies, techniques, and future directions,” *J. Electr. Syst.* **20**, 2030–2046 (2024).
6. O. O. Amoo, F. Osasona, A. Atadoga, B. S. Ayinla, O. A. Farayola, and T. O. Abrahams, “Cybersecurity threats in the age of IoT: A review of protective measures,” *Int. J. Sci. Res. Arch.* ***11****, 1304–1310 (2024).*
7. V. P. Gupta, “Smart sensors and Industrial IoT (IIoT): A driver of the growth of Industry 4.0,” in *Smart Sensors for Industrial Internet of Things: Challenges, Solutions and Applications*, (Springer, Cham, 2021), pp. 37–49.
8. A. Munshi, N. A. Alqarni, and N. Abdullah Almalki, “DDoS attack on IoT devices,” in *Proc. Int. Conf. Comput. Appl. Inf. Secur. (ICCAIS)* (IEEE, 2020), pp. 1–5.
9. M. Khatkar, K. Kumar, and B. Kumar, “An overview of distributed denial of service and internet of things in healthcare devices,” in *Proc. Res. Innov. Knowl. Manag. Technol. Appl. Bus. Sustain. (INBUSH)* (IEEE, 2020), pp. 44–48.
10. M. Thankappan, H. Rifà-Pous, and C. Garrigues, “Multi-Channel Man-in-the-Middle Attacks Against Protected Wi-Fi Networks: A State of the Art Review,” *Expert Syst. Appl.* **210**, 118401 (2022).
11. C. S. Kalutharage, X. Liu, C. Chrysoulas, N. Pitropakis, and P. Papadopoulos, “Explainable AI-based DDoS attack identification method for IoT networks,” *Computers* **12**, 32 (2023).
12. A. B. Sultan, S. Mehmood, and H. Zahid, “Man-in-the-middle attack detection for MQTT-based IoT devices using different machine learning algorithms,” in *Proc. Int. Conf. Artif. Intell. (ICAI)* (IEEE, 2022), pp. 118–121.
13. T. Horák and L. Huraj, “Smart thermostat as a part of IoT attack,” in *Advances in Intelligent Systems and Computing* (2019), pp. 156–163.
14. G. E. K. Raju, “Chip off IoT devices: Attacks and mitigations,” *Int. J. Recent Technol. Eng.* ***8****, 164–168 (2019).*
15. J. Yoon, “Deep-learning approach to attack handling of IoT devices using IoT-enabled network services,” *Internet Things* **11**, 100241 (2020).
16. D. Díaz López, M. Blanco Uribe, C. Santiago Cely, A. Vega Torres, N. Moreno Guataquira, S. Morón Castro, P. Nespoli, and F. Gómez Mármol, “Shielding IoT against cyber-attacks: An event-based approach using SIEM,” *Wirel. Commun. Mob. Comput.* ***2018****, 1–18 (2018).*