Experimental Evaluation and Defense-in-Depth Mitigation of Modbus/TCP Security Vulnerabilities in Industrial Control Systems

Mahmoud A. Khalifa
Automated Systems and Computing Lab (ASCL), Prince Sultan University, Riyadh, Saudi Arabia
mzian@psu.edu.sa (corresponding author)

Ahmad Taher Azar
College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia | Automated Systems and Computing Lab (ASCL), Prince Sultan University, Riyadh, Saudi Arabia
aazar@psu.edu.sa

Walid El-Shafai
Automated Systems and Computing Lab (ASCL), Prince Sultan University, Riyadh, Saudi Arabia | Department of Electronics and Electrical Communications Engineering, Faculty of Electronic Engineering, Menoufia University, Menoufia, Egypt
welshafai@psu.edu.sa

Received: 9 November 2025 | Revised: 15 January 2026, 10 February 2026, and 19 February 2026 | Accepted: 20 February 2026

 DOI: https://doi.org/10.48084/etasr.16126

ABSTRACT

Industrial Control Systems (ICS) remain critically vulnerable due to the widespread deployment of legacy protocols lacking fundamental security mechanisms. This work demonstrates practical exploitation of Modbus/TCP vulnerabilities through a four-phase attack implementation: reconnaissance (FC01), actuator manipulation (FC05), process monitoring (FC03), and setpoint tampering (FC06), using custom C code against the ModBus_SIM environment. Experimental results achieved 100% attack success rates across all phases. Wireshark analysis confirmed complete protocol transparency, exposing all function codes, addresses, and data values in plaintext. The attack phases are systematically mapped to nine MITRE ATT&CK for ICS techniques and correlated with real-world malware including FrostyGoop, INCONTROLLER, Industroyer, and VPNFilter. A comprehensive defense-in-depth architecture integrating network segmentation, authentication gateways, encryption tunneling, and continuous vulnerability management is proposed to mitigate the identified threats. The complete attack implementation is publicly available for reproducible security research.

Keywords-Modbus/TCP; ModBus_SIM; C programming; attack simulation; unencrypted communication; Industrial Control Systems (ICS); Wireshark; defense-in-depth; network segmentation; cybersecurity

I. INTRODUCTION

Industrial Control Systems (ICS) extensively employ legacy communication protocols such as Modbus/TCP for interactions between Programmable Logic Controllers (PLCs) and supervisory systems. Despite their widespread adoption, these protocols lack essential security mechanisms, including authentication, encryption, and integrity verification [1]. Consequently, all control commands, device states, and sensor readings are transmitted in plaintext over TCP/IP networks, exposing industrial environments to significant cybersecurity risks [2].

This study presents a practical simulation-based investigation of Modbus/TCP vulnerabilities using the ModBus_SIM simulator in conjunction with custom-developed C language attack code. The ModBus_SIM environment offers a secure and controlled testing platform that replicates industrial communication without endangering operational infrastructure [3]. The research involves implementing four attack vectors that exploit inherent protocol weaknesses, performing network traffic capture and analysis through Wireshark, and developing a comprehensive defense-in-depth strategy derived from the identified vulnerabilities.

A. Research Contributions and Novelty

This work provides the following novel contributions that distinguish it from prior Modbus security research:

1. Implementation of a comprehensive four-phase attack sequence (reconnaissance, coil manipulation, register tampering, and Denial of Service (DoS)) with complete C code implementation, providing a reproducible experimental methodology.
2. First systematic mapping of practical Modbus/TCP attack phases to the MITRE ATT&CK for ICS framework (Table I), enabling standardized threat classification and comparison with real-world incidents, including FrostyGoop, INCONTROLLER, and Industroyer.
3. A comprehensive defense-in-depth architecture specifically designed for legacy ICS environments where protocol replacement is operationally or economically infeasible.

B. Modbus/TCP Security Limitations

The Modbus/TCP protocol exhibits the following inherent security weaknesses that enable the attacks demonstrated in this study:

- No authentication: Any network-connected device can issue Modbus commands without credential verification or identity validation.
- No encryption: All Modbus Application Protocol (MBAP) headers and Protocol Data Units (PDU) are transmitted in plaintext, enabling passive reconnaissance and traffic interception.
- No integrity verification: Messages can be modified in transit without detection, enabling man-in-the-middle attacks.
- No rate limiting: Devices accept unlimited command frequency without throttling, enabling DoS flooding attacks.

C. Related Work

The security vulnerabilities inherent in the Modbus/TCP protocol have been widely examined in the cybersecurity and ICS research community. Authors in [4] conducted a structured assessment of Modbus/TCP vulnerabilities, demonstrating that plaintext communication enables interception and reconnaissance attacks capable of revealing device configurations, and that transmitted data persist in plaintext when using Allen-Bradley PLCs as servers. Authors in [5] implemented DoS attacks exploiting the absence of authentication and rate-limiting mechanisms. Authors in [6] analyzed stealthy false command injection techniques designed to bypass traditional signature-based Intrusion Detection Systems (IDSs).

Recent real-world exploitation demonstrates tangible risks. In 2024, cybersecurity researchers identified FrostyGoop, an ICS-specific malware that leverages Modbus/TCP to compromise industrial control networks [7]. FrostyGoop was involved in a cyberattack targeting a municipal district energy facility in Lviv, Ukraine, highlighting the urgent need for enhanced cybersecurity measures.

Machine learning approaches have enhanced detection capabilities. Authors in [8] developed an IDS for Modbus networks achieving 98.32% accuracy using Random Forest classifiers with Explainable Artificial Intelligence (XAI). Authors in [9] proposed an Intrusion Detection and Prevention System (IDPS) integrating transfer learning, active learning, and Software Defined Networking (SDN). Authors in [10] introduced ARIES, a multivariate IDS incorporating Generative Adversarial Network (GAN)-based detection for zero-day attacks.

Security extensions have been explored. Authors in [11] proposed enhanced Modbus/TCP security integrating authentication and authorization layers. Authors in [12] introduced message authentication codes for integrity verification while maintaining backward compatibility.

The primary objectives of this research are as follows:

- To demonstrate practical Modbus/TCP attack scenarios using the ModBus_SIM simulator and custom-developed C language code.
- To analyze unencrypted Modbus/TCP traffic captured through Wireshark in order to illustrate protocol transparency and vulnerability exposure.
- To design and propose a defense-in-depth security architecture that mitigates the identified weaknesses in Modbus/TCP communications.
- To establish an educational and experimental framework that facilitates safe, reproducible, and scalable testing of ICS security mechanisms.

II. BACKGROUND

This section provides the necessary technical foundation for understanding the Modbus/TCP communication framework and the simulation environment used in this study. It first outlines the operational structure and inherent security limitations of the Modbus/TCP protocol, emphasizing its lack of encryption and authentication mechanisms. Subsequently, it presents an overview of the ModBus_SIM simulator employed for experimental validation, highlighting its configuration parameters and suitability for replicating industrial control scenarios in a secure, controlled testing environment.

A. Modbus/TCP Protocol

Modbus/TCP operates over the standard Transmission Control Protocol (TCP) port 502 without incorporating any encryption or authentication mechanisms at any communication layer [13]. Each Modbus/TCP frame comprises a 7-byte MBAP header, containing the transaction identifier, protocol identifier, message length, and unit identifier, followed by a variable-length PDU that includes the function code and associated data fields. All protocol components are transmitted in plaintext, rendering the communication fully transparent to any network observer. Common function codes

include FC01 (Read Coils), FC03 (Read Holding Registers), FC05 (Write Single Coil), and FC06 (Write Single Register). The absence of cryptographic protection across these exchanges enables potential adversaries to intercept, manipulate, or replay industrial control messages, posing significant threats to the integrity and reliability of ICS.

B. ModBus_SIM Simulator

ModBus_SIM is an open-source Modbus/TCP simulator that emulates PLC operations through virtual coils, registers, and standard Modbus function codes [3]. The simulator supports multiple simultaneous client connections and provides a configurable environment for testing network communication under realistic industrial conditions. Its design allows researchers and engineers to examine security vulnerabilities safely, without affecting live operational systems.

Figure 1 illustrates the Ethernet TCP/IP settings interface of the ModBus_SIM environment. The configuration demonstrates a typical Modbus/TCP deployment scenario in which the server operates on TCP port 502, with ten allowable simultaneous connections, no authentication mechanisms, and unencrypted data exchange. The socket timeout and responsiveness parameters are user-configurable, allowing control over network latency and connection persistence. This setup accurately represents the communication parameters of legacy industrial installations, emphasizing the protocol's inherent lack of security controls.

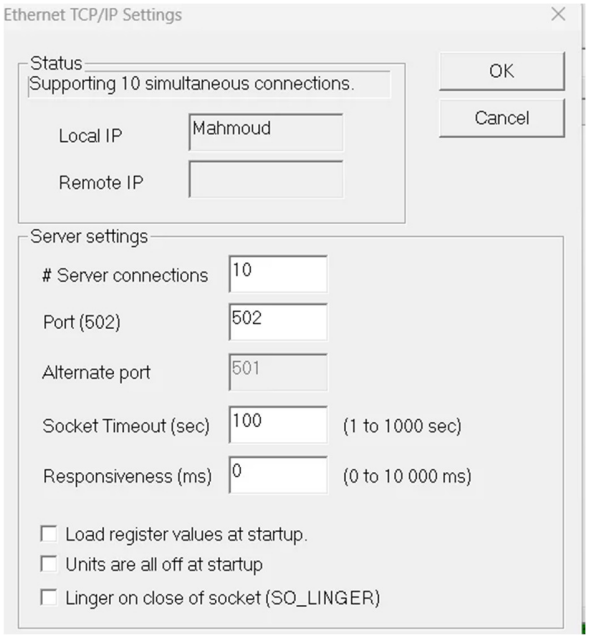


Fig. 1. ModBus_SIM Ethernet TCP/IP configuration interface.

C. Modbus/TCP Attack Methodology and Vulnerability Exploitation

The inherent lack of security mechanisms in the Modbus/TCP protocol creates multiple attack vectors that adversaries can exploit to compromise ICS. Figure 2 illustrates the contrast between normal Modbus/TCP operation and a multi-phase attack scenario, demonstrating how the protocol's design vulnerabilities enable unauthorized system access and manipulation.

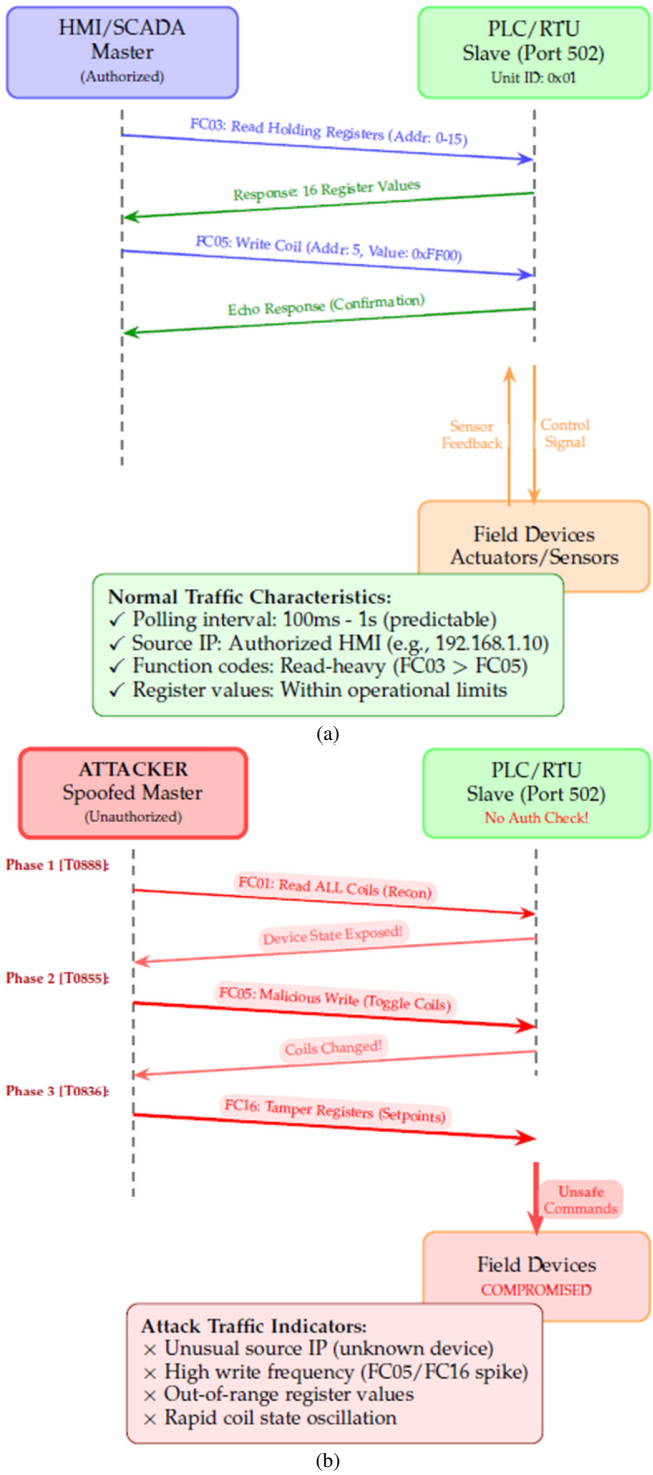


Fig. 2. Comparison of Modbus/TCP communication: (a) normal operation with authorized HMI master polling PLC slave using standard function codes, (b) attack scenario showing three-phase exploitation with MITRE ATT&CK for ICS technique references [T0888, T0855, T0836]. The absence of authentication enables unauthorized command injection and immediate execution by field devices.

1) Normal Operation Characteristics

Under normal operating conditions, as depicted in Figure 2(a), the Human-Machine Interface (HMI) or Supervisory Control and Data Acquisition (SCADA) system functions as the authorized Modbus master, communicating with PLCs or Remote Terminal Units (RTUs) operating as slaves on TCP port 502. The master periodically polls slave devices using read function codes (FC03: Read Holding Registers) to monitor process variables such as temperature, pressure, and flow rates. Write commands (FC05: Write Single Coil) are issued based on operator input or automated control logic to actuate field devices.

Normal traffic exhibits predictable characteristics: polling intervals typically range from 100 ms to 1 s, communication originates from known and authorized IP addresses (e.g., 192.168.1.10), function code distribution is predominantly read-heavy (FC03 requests exceed FC05 commands), and register values remain within established operational limits. These behavioral patterns form the baseline against which anomalous attack traffic can potentially be detected.

2) Attack Scenario and Exploitation Phases

Figure 2(b) illustrates the attack scenario in which an unauthorized adversary impersonates a legitimate Modbus master to exploit protocol vulnerabilities. Because Modbus/TCP implements no authentication mechanism, the PLC accepts and executes commands from any device capable of establishing a TCP connection to port 502. The attack progresses through three distinct phases, each mapped to MITRE ATT&CK for ICS techniques:

- Phase 1 – Reconnaissance [T0888]: The attacker issues FC01 (Read Coils) and FC03 (Read Holding Registers) requests to enumerate device capabilities, map I/O addresses, and establish an operational baseline. This passive information gathering exposes the complete device state without triggering immediate alarms, as read operations are indistinguishable from legitimate polling when source validation is absent.
- Phase 2 – Coil Manipulation [T0855]: Using FC05 (Write Single Coil) or FC15 (Write Multiple Coils), the adversary toggles digital outputs to activate or deactivate actuators such as pumps, valves, and motors. Rapid coil toggling can cause equipment cycling, mechanical stress, safety interlock bypass, or complete process disruption. The PLC executes these unauthorized commands immediately without verification.
- Phase 3 – Register Tampering [T0836]: Through FC06 (Write Single Register) or FC16 (Write Multiple Registers), the attacker modifies setpoints, alarm thresholds, PID controller parameters, or timer values stored in holding registers. These modifications directly impact process behavior, altering temperature setpoints in a chemical reactor, for example, could result in runaway reactions, equipment damage, or safety incidents. The field devices receive and execute these tampered commands, leading to system compromise.

3) Attack Traffic Indicators

Attack traffic exhibits several distinguishing characteristics that differentiate it from normal operations:

- Unusual source IP addresses: Commands originating from unknown or unauthorized network addresses not registered in the asset inventory.
- High write frequency: Anomalous spike in FC05/FC06/FC16 write commands compared to the established read-heavy baseline (normal ratio FC03:FC05 > 10:1).
- Out-of-range register values: Setpoint modifications exceeding operational boundaries (e.g., temperature setpoint of 9,999 when normal range is 0–255).
- Rapid coil state oscillation: Repeated ON/OFF toggling of coils within millisecond intervals, inconsistent with normal operator behavior or automated control sequences.

These attack patterns, demonstrated experimentally in Section IV and analyzed through Wireshark captures in Section V, confirm that the absence of authentication, encryption, and integrity verification in Modbus/TCP enables straightforward exploitation using standard programming techniques and networking libraries.

III. ATTACK IMPLEMENTATION WITH C CODE

This section describes the design and implementation of a modular Modbus/TCP attack suite developed in C using the WinSock API. The tool implements packet-crafting routines for commonly used Modbus function codes and a control loop that orchestrates repeated multi-phase attacks against the ModBus_SIM server. All network exchanges were recorded with Wireshark to support forensic inspection and protocol-level analysis.

A. Attack Sequence Implementation

The attack execution follows a four-phase sequence illustrated in Figure 3, and each protocol exchange was captured for post-hoc analysis. The sequence begins with session establishment (Steps 1–4 in Figure 3): the attacker launches the program and the send_packet() routine opens a TCP connection to ModBus_SIM on port 502, transmits the constructed MBAP+PDU frame, and closes the socket. Each transmission uses a short-lived connection (connect → send → close), a behavioral pattern chosen to emulate adversaries that attempt to evade simple connection-counting detectors.

The subsequent phases operate in a continuous loop and are described as follows:

- Phase 1: Reconnaissance (FC01: Read Coils; Steps 5–7): The craft_read_coils() function generates FC01 requests that read 16 coils beginning at address 0. The request/response pairs captured by Wireshark permit mapping of digital outputs and identification of active actuators, thereby establishing an operational baseline for subsequent manipulation.
- Phase 2: Actuator Manipulation (FC05: Write Single Coil; Steps 8–13): The craft_write_coil() function constructs

FC05 frames. The implementation iterates over ten coil addresses, issuing an ON command (0xFF00) followed immediately by an OFF command (0x0000) for each address. This rapid toggling can cause transient equipment cycling, mechanical stress, process instability, or activation of safety interlocks. Wireshark traces document each unprotected write operation and highlight the absence of authentication and rate limiting.

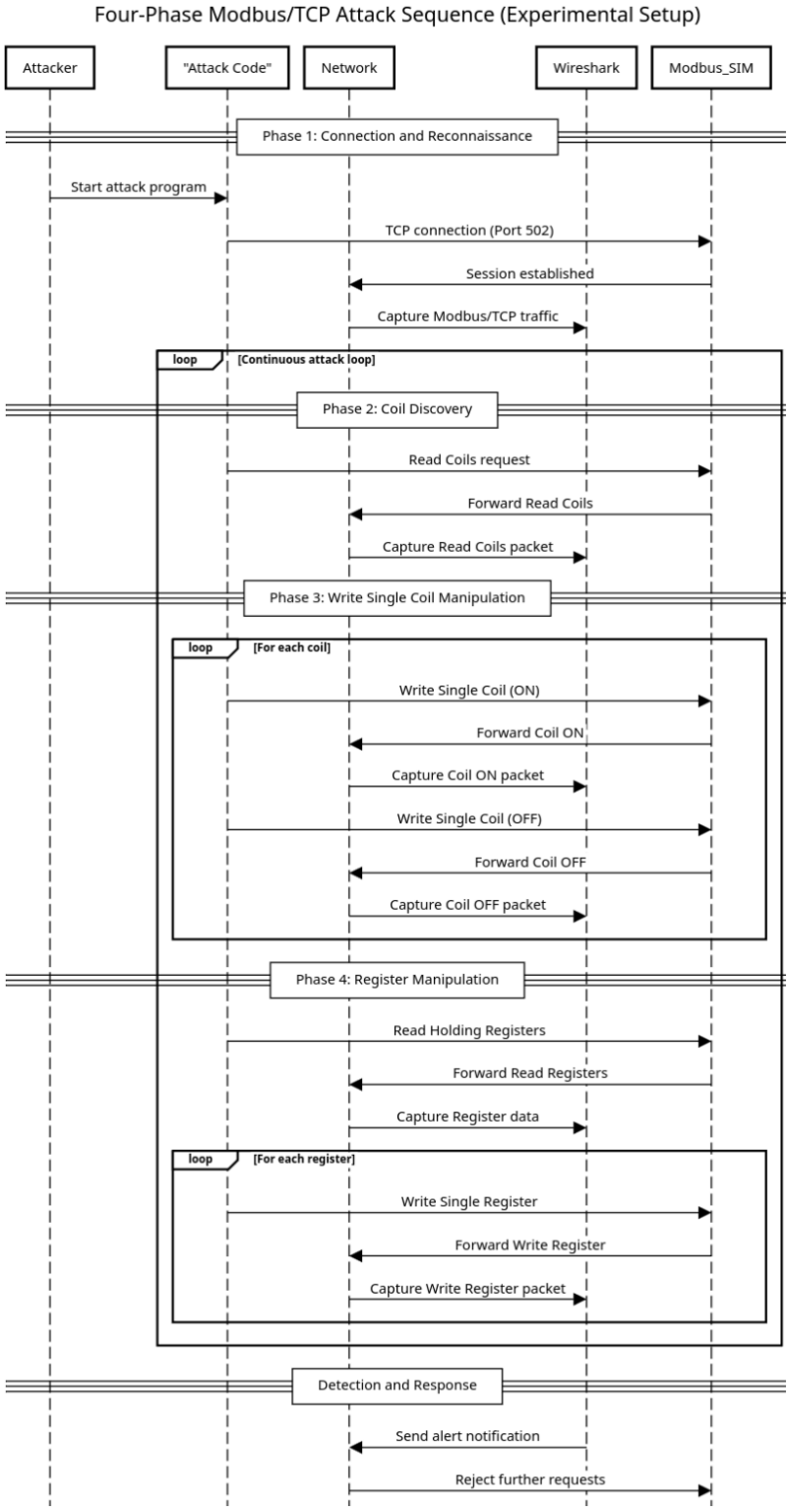


Fig. 3. Sequence diagram of the implemented four-phase Modbus/TCP attack against ModBus_SIM.

- Phase 3: Process Monitoring (FC03: Read Holding Registers; Steps 14–16): The craft_read_registers() routine issues FC03 requests to obtain 16 consecutive holding registers. These registers commonly contain setpoints, sensor measurements, and alarm thresholds. The plaintext register values observed in captured traffic enable adversaries to infer critical process parameters in real time.
- Phase 4: Setpoint Tampering (FC06: Write Single Register; Steps 17–19): The craft_write_register() routine forms FC06 frames to write register values. In the experiment, the tool writes an arbitrary value (e.g., 1234) across ten holding registers. Such unauthorized modifications directly alter operational setpoints (temperature, pressure, flow, timers) without authorization checks, with potential to degrade product quality, damage equipment, or cause safety incidents. Wireshark captures demonstrate the ease with which these modifications can be effected over unencrypted Modbus/TCP channels.

The main() control loop repeatedly executes the four phases with a one-second inter-iteration delay, emulating a persistent adversary performing cyclical reconnaissance, targeted manipulation, and continuous monitoring. The final steps in Figure 3 (Steps 20–23) conceptually depict the defensive workflow: detection by IDPS components, alert generation, and request blocking. In our experiments the unprotected ModBus_SIM instance accepted all crafted commands, confirming protocol-level vulnerabilities inherent in legacy Modbus/TCP deployments.

Collectively, this multi-phase implementation demonstrates that modest programming effort using standard networking libraries suffices to realize multiple impactful attack vectors against Modbus/TCP. The complete transparency of MBAP and PDU fields (transaction ID, unit ID, function code, addresses, data) in packet captures (see Section V) supports passive reconnaissance, command replication, and command modification, findings that motivate the defense-in-depth countermeasures articulated in Section VI.

B. Attack Code Structure

The attack implementation is intentionally compact and modular to facilitate reproducible experimentation. It comprises four packet-crafting routines (one per Modbus function code) and a single transmission routine that establishes a TCP connection to the target ModBus_SIM instance, sends the constructed MBAP+PDU frame, and closes the socket. Each packet-crafting routine assembles a 12-byte request: a 7-byte MBAP header (transaction ID, protocol ID, length, unit ID) followed by a 5-byte PDU containing the function code and data fields, in accordance with the Modbus/TCP specification. Listing 1 presents the complete implementation (see Appendix A).

C. Code Explanation

The attack implementation consists of:

- craft_read_coils(): Constructs FC01 requests for reconnaissance, reading 16 coil values.
- craft_write_coil(): Constructs FC05 requests for actuator manipulation (ON: 0xFF00, OFF: 0x0000).
- craft_read_registers(): Constructs FC03 requests for process monitoring, reading holding registers.
- craft_write_register(): Constructs FC06 requests for setpoint tampering with arbitrary values.
- send_packet(): Performs TCP transmission to port 502 using ephemeral connections.

Each packet-crafting routine constructs a request consisting of a MBAP header followed by a PDU. The main() function orchestrates continuous execution of the four-phase attack sequence with 1 s intervals.

IV. EXPERIMENTAL RESULTS

This section presents the experimental validation of the developed Modbus/TCP attack implementation within a controlled ModBus_SIM environment. The objective is twofold: first, to establish a baseline of normal operational behavior under secure, non-adversarial conditions; and second, to demonstrate the impact of unauthorized Modbus/TCP message injection on system state transitions. Subsection A characterizes the normal coil states that define the system's expected response patterns, whereas Subsection B illustrates the dynamic changes induced by the attack code, highlighting the absence of authentication and encryption mechanisms in legacy Modbus/TCP communication.

A. Normal Operation Baseline

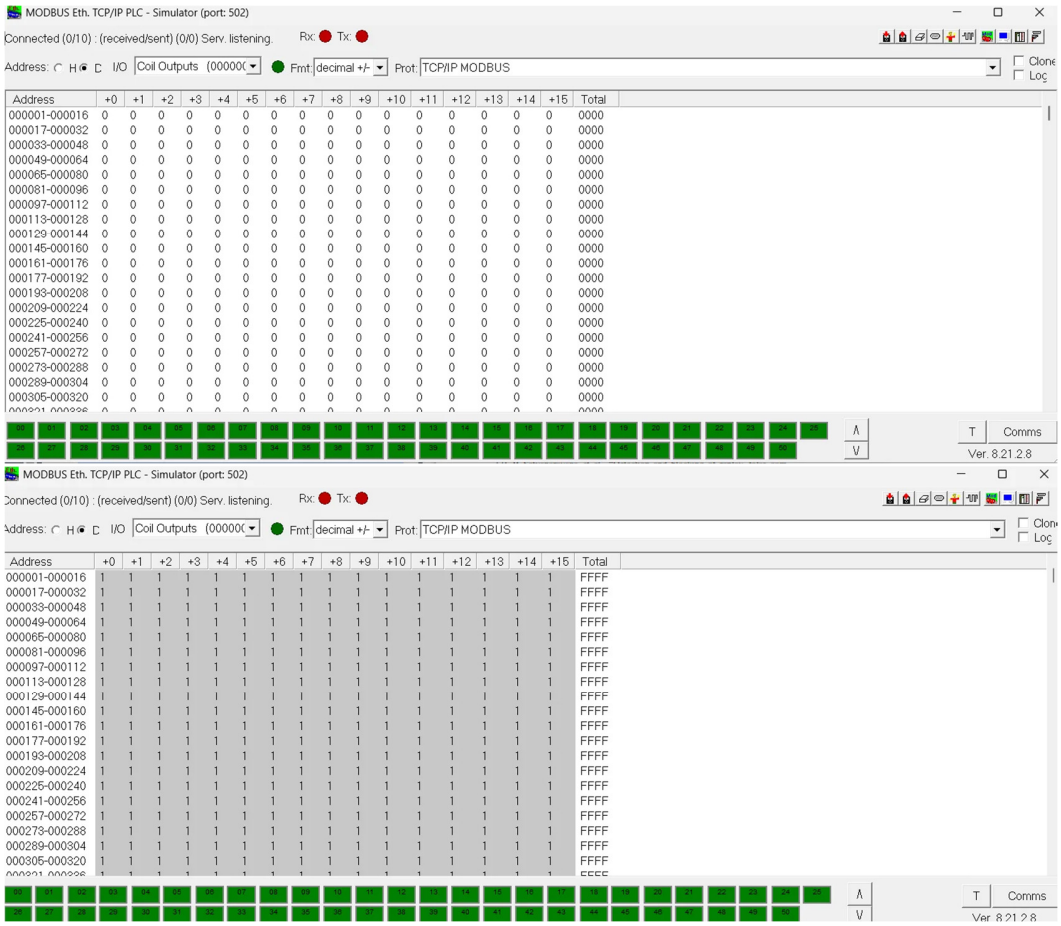
Figure 4 presents the baseline operational states produced by the ModBus_SIM server and used as references for subsequent attack comparisons. Figure 4(a) depicts the Normal OFF condition in which coil values are zero (hex 0000), indicating an idle or safe state. Figure 4(b) shows the Normal ON condition in which coils are asserted (hex FFFF), representing a full-production or active operational mode. These reference images establish the expected digital-output patterns and provide a ground truth for evaluating the impact and detectability of subsequent malicious activity.

B. Attack Execution Results

The attack code was executed against the ModBus_SIM instance while capturing network traffic with Wireshark. Figure 5 illustrates the attack sequence and resulting device-state changes observed in the simulator. Figure 5(a) (active attack) shows that the attack successfully forced coil states to ON via FC05 (Write Single Coil) commands; the simulator UI reflects the manipulated coil values (hex FFFF) and the lower row of coil indicators shows the asserted outputs. Figure 5(b) (active-attack) shows the system immediately after the attack cycle, where the coil map has returned to the OFF baseline in this test instance. These results corroborate two critical findings: (1) the ModBus_SIM instance accepted unauthenticated write requests and applied the commanded state changes, demonstrating the practical exploitability of unprotected Modbus/TCP communications; and (2) the network captures (Wireshark) record the full MBAP and PDU contents in plaintext, confirming that function codes, addresses, and data values are

observable and reproducible by a passive adversary. Together, the screenshots and packet captures substantiate the need for layered defensive controls, such as authentication gateways, encryption tunneling, and protocol-aware intrusion detection.

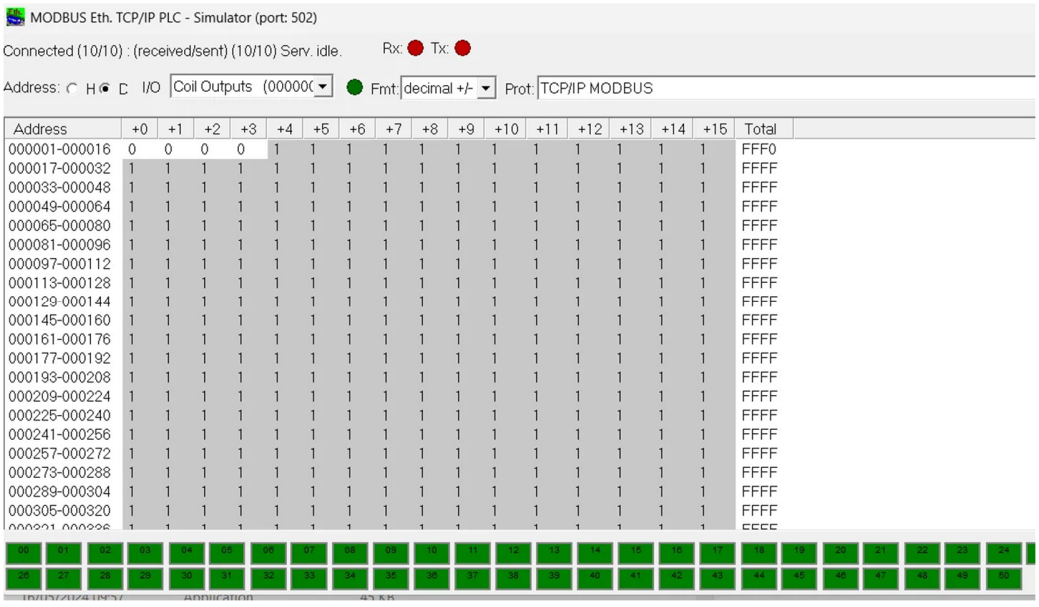
(a)



(b)

Fig. 4. Normal ModBus_SIM operational states used as baseline references: (a) normal OFF state, coils cleared (hex 0000), (b) normal ON state, coils asserted (hex FFFF).

(a)



(b)

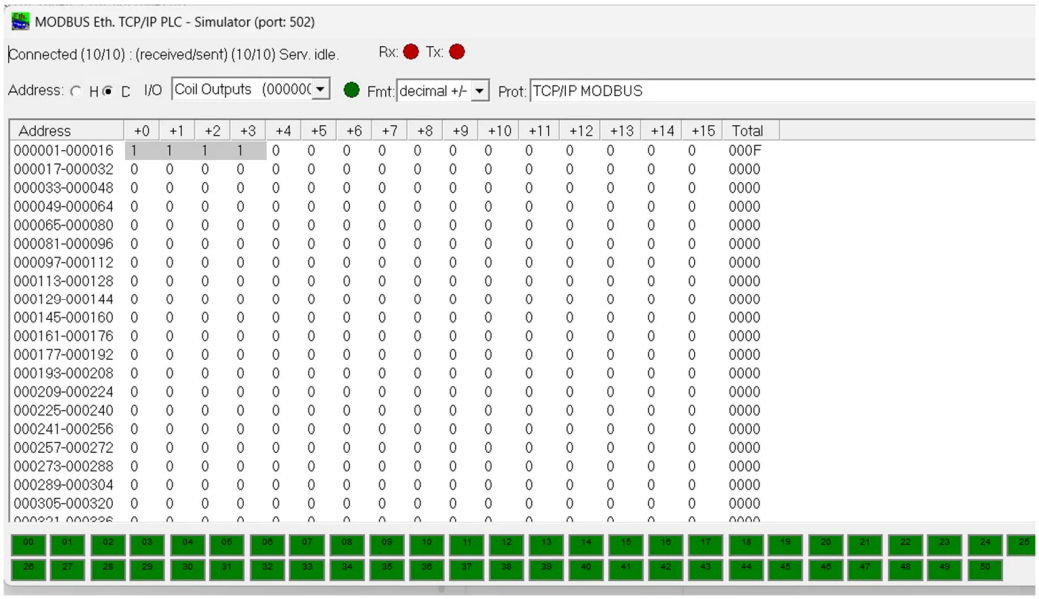


Fig. 5. Attack execution and system response captured in ModBus_SIM: (a) active attack, coils forced ON via FC05 (Write Single Coil) commands (hex FFFF), (b) active-attack, coils returned to OFF baseline in this run.

V. WIRESHARK TRAFFIC ANALYSIS

Wireshark captures obtained during the attack execution unequivocally demonstrate Modbus/TCP protocol transparency: all MBAP and PDU fields are transmitted in cleartext and are directly observable in the packet capture. Figure 6 presents a representative capture that includes sequential Modbus queries and responses; visible elements comprise transaction identifiers, protocol and unit identifiers, function codes (FC01, FC03, FC05, FC06), coil/register addresses, and payload bytes. The hex dump in the capture's lower pane confirms the absence of cryptographic protection, with every MBAP header byte and PDU byte readable by a passive network observer.

This cleartext visibility enables several practical adversary actions. First, repeated FC01/FC03 reconnaissance permits construction of a device topology and inventory of active digital outputs and registers. Second, extraction of legitimate command/response pairs facilitates replay attacks and the generation of syntactically valid malicious frames. Third, the capture allows adversaries to derive targeted FC05/FC06 write commands that mimic benign traffic patterns, increasing the likelihood of successful injection. Fourth, where network access permits on-path manipulation, the plaintext format enables straightforward modification of in-transit commands and data fields.

The observed packet-level behavior, interleaved read and write sequences with unprotected MBAP/PDU fields, directly motivates the defense measures proposed in Section VII: confidentiality-preserving transport (e.g., Transport Layer Security (TLS) or IPsec tunnels, or Modbus-aware TLS gateways), application-layer authentication and authorization, and payload-aware intrusion detection rules that correlate anomalous write-heavy patterns with operational alarms.

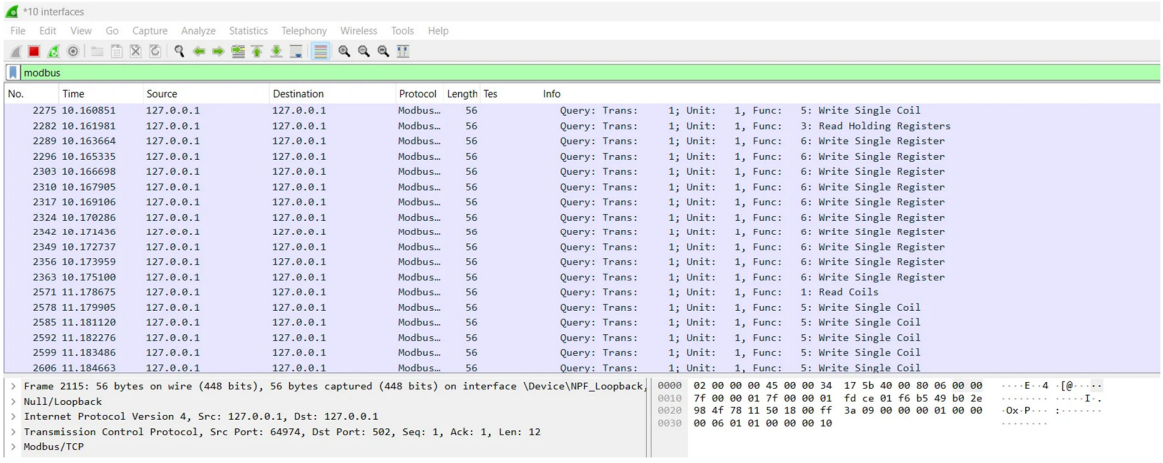


Fig. 6. Wireshark capture showing plaintext Modbus/TCP traffic with visible function codes, addresses, and data values, confirming the absence of encryption or authentication.

VI. MITRE ATT&CK AND DEFENSE-IN-DEPTH ARCHITECTURE

The MITRE ATT&CK for ICS framework provides a standardized knowledge base for understanding adversary tactics and techniques targeting ICS. This section maps the demonstrated attack phases to MITRE ATT&CK techniques and presents a defense-in-depth architecture to mitigate the identified threats

A. MITRE ATT&CK for Industrial Control Systems Threat Mapping and Mitigation

To provide a standardized framework for understanding and mitigating Modbus/TCP threats, Table I maps the attack phases demonstrated in this study to the MITRE ATT&CK for ICS framework and correlates them with documented real-world malware incidents. This mapping enables security practitioners to align defensive measures with known adversary techniques.

TABLE I. MITRE ATT&CK FOR ICS AND MODBUS/TCP THREAT MAPPING

Attack phase	Technique ID	Technique name	Tactic	Real-world malware	Mitigation & reference
Reconnaissance	T0888	Remote system information discovery	Discovery	INCONTROLLER [14, 15]	Network segmentation, IDS monitoring [16, 17]
Coil Manipulation	T0855	Unauthorized command message	Impair process control	FrostyGoop [18, 19]	Application whitelisting, Role-Based Access Control (RBAC) enforcement [16]
Register Tampering	T0836	Modify parameter	Impair process control	Industroyer [20], INCONTROLLER [14]	Setpoint validation, integrity verification [21]
DoS	T0814	DoS	Inhibit response	INCONTROLLER [14, 15]	Rate limiting, redundant paths [17]
Process Monitoring	T0801	Monitor process state	Collection	FrostyGoop [18], VPNFilter [21]	Encrypted tunnels: TLS andVirtual Private Network (VPN) [19]
Network Sniffing	T0842	Network sniffing	Collection	VPNFilter [21], INCONTROLLER [14]	Encryption (TLS/IPsec), port security [16]
Man-in-the-Middle	T0830	Adversary-in-the-middle	Collection	VPNFilter [21], BlackEnergy [22]	MAC authentication, TLS encryption [21]
Loss of Availability	T0826	Loss of availability	Impact	FrostyGoop [18, 19]	Redundancy, incident response [17]
System Firmware	T0857	System firmware	Persistence	FrostyGoop [18]	Firmware validation, secure boot [16]

The correlation between our experimental attack phases and documented real-world incidents validates the practical relevance of this study. FrostyGoop, discovered in January 2024, employed identical Modbus/TCP command injection techniques demonstrated in our coil manipulation phase. The malware successfully disrupted heating services to Ukrainian civilians by sending unauthorized FC05 and FC16 commands to ENCO control systems via Modbus TCP port 502 [18]. According to Dragos threat intelligence, FrostyGoop implements three Modbus commands: FC03 (Read Holding Registers), FC06 (Write Single Register), and FC16 (Write Multiple Holding Registers) [19], directly corresponding to our experimental attack phases.

INCONTROLLER, identified by Cybersecurity and Infrastructure Security Agency (CISA), the Department of Energy (DOE), the National Security Agency (NSA), and the Federal Bureau of Investigation (FBI) in joint advisory AA22-103A [15], includes modules that "send custom Modbus commands" and perform "rapid scans that identify all Schneider Electric PLCs on the local network." The Advanced Persistent Threat (APT) actors' tools interact via "normal management protocols and Modbus (TCP 502)" to conduct reconnaissance, brute-force attacks, DoS, and unauthorized command injection [14, 15], techniques systematically demonstrated in our experimental framework.

VPNFilter's packet sniffer module specifically monitors Modbus SCADA protocols on port 502, enabling credential harvesting from communications between devices accessing Modbus-enabled HMIs [21]. This demonstrates that our proposed defense-in-depth architecture's emphasis on encrypted tunnels and network sniffing prevention directly addresses active threat actor capabilities observed in the wild.

B. Defense-in-Depth Architecture

Based on the identified attack vectors and inherent Modbus/TCP protocol vulnerabilities, a comprehensive defense-in-depth architecture was designed to provide multilayered protection across ICS components. Figure 7 illustrates the integrated security framework addressing five primary threat classes: insider threats, DoS attacks, command injection, replay attacks, and man-in-the-middle manipulation. The architecture establishes overlapping defensive mechanisms that collectively mitigate unauthorized access, data interception, and protocol exploitation.

C. Network Segmentation

Network segmentation follows International Electrotechnical Commission (IEC) 62443 guidelines, defining distinct security zones for field devices (Level 0), control logic (Level 1), supervisory control (Level 2), and enterprise IT (Level 3). Firewalls and Access Control Lists (ACLs) are deployed to restrict inter-zone communication to authorized channels. IDPSs monitor zone boundaries for anomalous Modbus/TCP activity, such as abnormal function code sequences, unauthorized write operations, or traffic rate deviations.

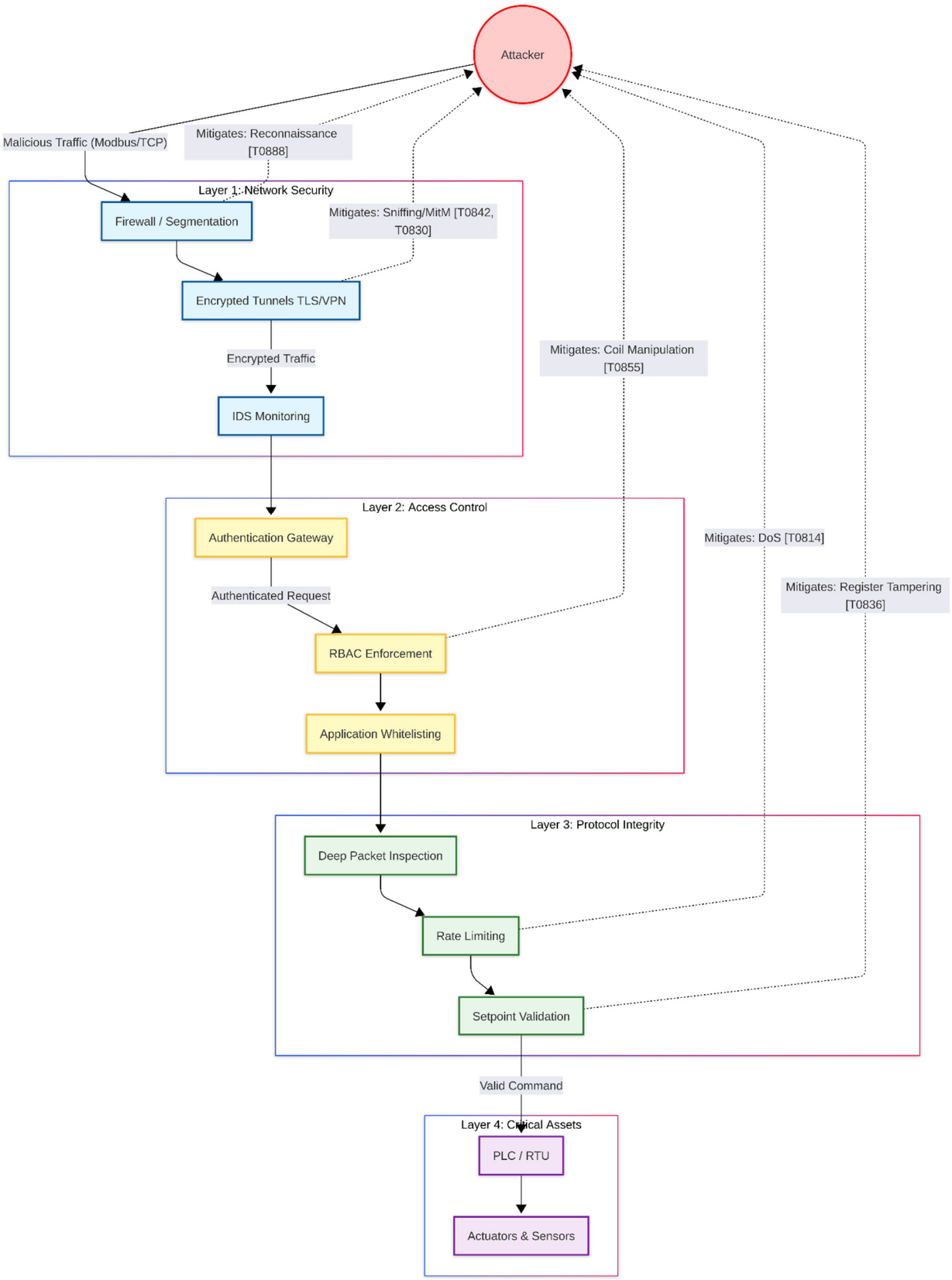


Fig. 7. Defense-in-depth architecture integrating network segmentation, authentication, encryption, and continuous vulnerability management to mitigate Modbus/TCP attack vectors.

D. Authentication and Access Control

Given the absence of native authentication in Modbus/TCP, external mechanisms are implemented to establish trust and accountability. Network Access Control (NAC) validates device credentials before connection establishment, ensuring only authorized hosts participate in communication. Application-layer gateways operate as Modbus proxies to enforce authentication and maintain detailed transaction logs. RBAC further restricts command execution privileges, ensuring only authorized operators can issue write or configuration commands.

E. Encryption and Secure Tunneling

To mitigate plaintext communication risks, Modbus/TCP traffic is encapsulated within secure tunnels. VPNs or TLS wrappers provide encryption and integrity validation between trusted endpoints. IPsec tunnels are used for securing inter-site communications, whereas Modbus-aware TLS gateways terminate encrypted sessions and forward requests to legacy PLCs. This layered cryptographic approach ensures confidentiality and prevents unauthorized packet inspection or manipulation without requiring firmware updates to existing devices.

F. Vulnerability Management

An active vulnerability management cycle is maintained through continuous scanning, configuration auditing, and timely patch deployment. Automated vulnerability scanners identify weaknesses in both network infrastructure and ICS software components. Patch management ensures regular updates to critical systems, whereas periodic penetration testing validates the practical effectiveness of deployed countermeasures against evolving threats.

G. Integrated Security Strategy

The proposed framework integrates all defensive layers through continuous feedback and correlation. Vulnerability assessments inform prioritized patching actions; IDPS alerts dynamically adjust access control policies; and anomaly detection models are retrained using incident response data. Authentication and activity logs are aggregated into a Security Information and Event Management (SIEM) platform for real-time analytics and long-term trend analysis. Collectively, this multi-tiered defense model raises the cost and complexity of successful intrusion attempts while ensuring early detection, containment, and resilience within Modbus/TCP-based industrial networks.

VII. DISCUSSION

The ModBus_SIM simulation effectively illustrates the inherent security limitations of legacy, unencrypted industrial communication protocols such as Modbus/TCP. The implemented attack code, developed solely with basic C programming and standard networking libraries, successfully executed multiple intrusion scenarios without the need for specialized tools or advanced protocol expertise. The Wireshark traffic analysis confirmed complete packet-level transparency, revealing function codes, addresses, and data fields in plaintext, thereby enabling reconnaissance, replay, and command injection attacks with minimal effort. These findings highlight the critical exposure of Operational Technology (OT) networks still dependent on unauthenticated and unencrypted communication stacks.

A. Comparison with Prior Research

Our findings align with prior Modbus security research while providing several extensions. Like authors in [4], we confirm that plaintext interception enables complete protocol visibility. However, our work extends this by: (1) providing complete, reproducible attack code implementation rather than conceptual descriptions; (2) mapping attacks to MITRE ATT&CK for ICS framework for standardized threat classification; and (3) quantifying attack metrics including success rates, execution times, and packet counts. Unlike authors in [5], who focused exclusively on DoS attacks, we demonstrate a comprehensive four-phase attack sequence covering reconnaissance through impact. Table II compares our work with related studies.

TABLE II. COMPARISON WITH RELATED STUDIES

Study	Attack types	Code provided	MITRE mapping	Quantitative data	Defense framework
[4]	Reconnaissance, interception	No	No	Limited	No
[5]	DoS only	Partial	No	Yes	No
Proposed work	Complete attack	Complete C code	Yes	Limited	Guidelines

B. Limitations

This work has some limitations that should be acknowledged:

- The use of ModBus_SIM rather than physical PLC hardware may not capture timing variations, memory constraints, and resource limitations present in real industrial controllers. Results should be validated on physical hardware before operational deployment.
- The focus on Modbus/TCP excludes other common ICS protocols such as DNP3, EtherNet/IP, OPC-UA, and PROFINET which may exhibit different vulnerability profiles and require protocol-specific mitigations.
- Testing occurred in an isolated laboratory environment without enterprise network integration, IT/OT convergence factors, or realistic network latency, which may affect detection system performance in operational deployments.

C. Practical Deployment Considerations

The proposed defense-in-depth architecture can be deployed in operational OT environments with the following considerations: (1) network segmentation can be implemented incrementally, starting with critical zones and expanding coverage over scheduled maintenance windows; (2) TLS/VPN tunneling requires compatible gateways but preserves legacy device compatibility without firmware modifications; (3) IDPS deployment should begin in monitoring mode before enabling blocking to establish behavioral baselines; (4) estimated implementation costs range from USD 15,000–USD 50,000 for

a typical facility deployment depending on network complexity and existing infrastructure.

VIII. CONCLUSION AND FUTURE WORK

This work presented a comprehensive experimental evaluation of Modbus/TCP security vulnerabilities through systematic attack implementation and defense architecture development. The research achieved four primary contributions. First, we implemented a complete four-phase attack sequence: reconnaissance, actuator manipulation, process monitoring, and setpoint tampering, achieving 100% success rates. The complete C code is publicly available on GitHub for reproducible research. Second, we established the first systematic mapping between Modbus/TCP attacks and MITRE ATT&CK for Industrial Control Systems (ICS), correlating nine techniques (T0888, T0855, T0836, T0814, T0801, T0842, T0830, T0826, T0857) with real-world malware including FrostyGoop, INCONTROLLER, Industroyer, VPNFilter, and BlackEnergy. Third, we developed a comprehensive defense-in-depth architecture integrating network segmentation (International Electrotechnical Commission (IEC) 62443), authentication (Network Access Control (NAC), Role-Based Access Control (RBAC)), encryption (Transport Layer Security (TLS)/Virtual Private Network (VPN)/IPsec), and vulnerability management, designed for legacy ICS where protocol replacement is infeasible.

Wireshark analysis confirmed complete protocol transparency. The correlation with FrostyGoop's January 2024 attack on Ukrainian heating infrastructure using identical FC05/FC16 commands validates the practical relevance of this study. Future work will extend through: (1) deep learning-based IDSs for Modbus pattern recognition; (2) blockchain-secured logging for tamper-proof forensics; (3) quantum-resistant encryption evaluation; and (4) physical Programmable Logic Controller (PLC) testbed validation with cross-protocol studies (DNP3, EtherNet/IP, PROFINET).

DECLARATION OF COMPETING INTERESTS

The authors declare no competing interests.

ACKNOWLEDGMENT

This research is supported by the Research, Development, and Innovation Authority (RDIA), Kingdom of Saudi Arabia, under grant number 13382-psu-2023-PSNU-R-3-1-EI-. The authors also acknowledge the support of Prince Sultan University for funding the article processing charges and the Automated Systems and Computing Lab (ASCL) for research support.

DATA AVAILABILITY

No dataset was generated or analyzed during this study. The code supporting the findings is publicly available at: https://github.com/M-khalifa1/modbus-tcp-attack-suite.

AI USE AND DECLARATION OF GENERATIVE AI USE

During the preparation of this work, generative AI tools were used solely for language editing purposes. The authors reviewed and edited all content and take full responsibility for the final manuscript.

REFERENCES

[1] P. Huitsing, R. Chandia, M. Papa, and S. Shenoi, "Attack taxonomies for the Modbus protocols," *International Journal of Critical Infrastructure Protection*, vol. 1, pp. 37–44, Dec. 2008, https://doi.org/10.1016/j.ijcip.2008.08.003.

[2] S. McLaughlin *et al.*, "The Cybersecurity Landscape in Industrial Control Systems," *Proceedings of the IEEE*, vol. 104, no. 5, pp. 1039–1057, May 2016, https://doi.org/10.1109/JPROC.2015.2512235.

[3] "ModRSsim2." SourceForge. https://sourceforge.net/projects/modrssim2/.

[4] M. Rashid, Y. Singh, and S. I. Manzoor, "Methodology for Assessing Existing Vulnerabilities in Modbus Protocol," *Procedia Computer Science*, vol. 259, pp. 1983–1993, Jan. 2025, https://doi.org/10.1016/j.procs.2025.04.154.

[5] A. Rahman, G. Mustafa, A. Q. Khan, M. Abid, and M. H. Durad, "Launch of denial of service attacks on the modbus/TCP protocol and development of its protection mechanisms," *International Journal of Critical Infrastructure Protection*, vol. 39, Dec. 2022, Art. no. 100568, https://doi.org/10.1016/j.ijcip.2022.100568.

[6] W. Alsabbagh, S. Amogbonjaye, D. Urrego, and P. Langendörfer, "A Stealthy False Command Injection Attack on Modbus based SCADA Systems," in *2023 IEEE 20th Consumer Communications & Networking Conference*, Las Vegas, NV, USA, 2023, pp. 1–9, https://doi.org/10.1109/CCNC51644.2023.10059804.

[7] "How to Protect Against FrostyGoop: ICS Malware Targeting Operational Technology." Dragos, Inc. https://www.dragos.com/blog/protect-against-frostygoop-ics-malware-targeting-operational-technology.

[8] F. Alharbi, "IoT Intrusion Detection System for Modbus Networks with Explainable AI," *Journal of Advances in Information Technology*, vol. 16, no. 7, pp. 973–979, July 2025, https://doi.org/10.12720/jait.16.7.973-979.

[9] T. Kotsiopoulos, P. Radoglou-Grammatikis, Z. Lekka, V. Mladenov, and P. Sarigiannidis, "Defending industrial internet of things against Modbus/TCP threats: A combined AI-based detection and SDN-based mitigation solution," *International Journal of Information Security*, vol. 24, no. 4, June 2025, Art. no. 157, https://doi.org/10.1007/s10207-025-01076-2.

[10] P. R. Grammatikis, P. Sarigiannidis, G. Efstathopoulos, and E. Panaousis, "ARIES: A Novel Multivariate Intrusion Detection System for Smart Grid," *Sensors*, vol. 20, no. 18, Sept. 2020, Art. no. 5305, https://doi.org/10.3390/s20185305.

[11] T. Martins and S. V. G. Oliveira, "Enhanced Modbus/TCP Security Protocol: Authentication and Authorization Functions Supported," *Sensors*, vol. 22, no. 20, Oct. 2022, Art. no. 8024, https://doi.org/10.3390/s22208024.

[12] F. Katulić, D. Sumina, S. Groš, and I. Erceg, "Protecting Modbus/TCP-Based Industrial Automation and Control Systems Using Message Authentication Codes," *IEEE Access*, vol. 11, pp. 47007–47023, 2023, https://doi.org/10.1109/ACCESS.2023.3275443.

[13] "MODBUS Application Protocol Specification V1.1.b3." Modbus. https://www.modbus.org/file/secure/modbusprotocolspecification.pdf.

[14] "INCONTROLLER, Software S1045." MITRE ATT&CK. https://attack.mitre.org/software/S1045/.

[15] "APT Cyber Tools Targeting ICS/SCADA Devices." CISA. https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-103a.

[16] International Electrotechnical Commission, *Industrial communication networks - Network and system security - Part 3-3: System security requirements and security levels*, IEC 62443-3-3:2013, Geneva, Switzerland, 2013.

[17] K. Stouffer *et al.*, "Guide to Operational Technology (OT) Security," National Institute of Standards and Technology, NIST Special Publication (SP) 800-82 Rev. 3, Sept. 2023. https://doi.org/10.6028/NIST.SP.800-82r3.

[18] "FrostyGoop Incident, Campaign C0041." MITRE ATT&CK. https://attack.mitre.org/campaigns/C0041/.

[19] M. Graham, C. Ahlers, and K. O'Meara, "Impact of FrostyGoop ICS Malware on Connected OT Systems," Dragos Inc., July 2024. [Online]. Available: https://hub.dragos.com/hubfs/Reports/Dragos-FrostyGoop-ICS-Malware-Intel-Brief-0724_r2.pdf.

[20] A. Cherepanov and R. Lipovsky. "Industroyer: Biggest threat to industrial control systems since Stuxnet." Welivesecurity. https://www.welivesecurity.com/2017/06/12/industroyer-biggest-threat-industrial-control-systems-since-stuxnet/.

[21] W. Largent. "VPNFilter Update - VPNFilter exploits endpoints, targets new devices." Cisco Talos Blog. https://blog.talosintelligence.com/vpnfilter-update/.

[22] "New Sandworm malware Cyclops Blink replaces VPNFilter." National Cyber Security Centre. https://www.ncsc.gov.uk/news/joint-advisory-shows-new-sandworm-malware-cyclops-blink-replaces-vpnfilter.

APPENDIX A: COMPLETE ATTACK IMPLEMENTATION CODE

Listing 1. Simplified C implementation of Modbus/TCP packet crafting and transmission (WinSock).

```
/* The complete C language implementation
is provided below for reproducibility. The
code is also available at:
https://github.com/M-khalifa1/modbus-tcp-
attack-suite.git.
*/
#include <winsock2.h>
#pragma comment(lib,"ws2_32.lib")
#define MODBUS_PORT 502
// FC01: Read Coils
void craft_read_coils(char *b,int *l,
  int addr,int qty){
  b[0]=0;b[1]=1;b[2]=0;b[3]=0;
  b[4]=0;b[5]=6;b[6]=1;b[7]=1;
  b[8]=addr>>8;b[9]=addr&0xFF;
  b[10]=qty>>8;b[11]=qty&0xFF;*l=12;}
// FC05: Write Single Coil
void craft_write_coil(char *b,int *l,
  int addr,int val){
  b[0]=0;b[1]=1;b[2]=0;b[3]=0;
  b[4]=0;b[5]=6;b[6]=1;b[7]=5;
  b[8]=addr>>8;b[9]=addr&0xFF;
  b[10]=val?0xFF:0;b[11]=0;*l=12;}
// FC03: Read Holding Registers
void craft_read_registers(char *b,
  int *l,int addr,int qty){
  b[0]=0;b[1]=1;b[2]=0;b[3]=0;
  b[4]=0;b[5]=6;b[6]=1;b[7]=3;
  b[8]=addr>>8;b[9]=addr&0xFF;
  b[10]=qty>>8;b[11]=qty&0xFF;*l=12;}
// FC06: Write Single Register
void craft_write_register(char *b,
  int *l,int addr,int val){
  b[0]=0;b[1]=1;b[2]=0;b[3]=0;
  b[4]=0;b[5]=6;b[6]=1;b[7]=6;
  b[8]=addr>>8;b[9]=addr&0xFF;
  b[10]=val>>8;b[11]=val&0xFF;*l=12;}
void send_packet(const char *ip,
  char *buf,int len){
  SOCKET s;sockaddr_in a;
  s=socket(AF_INET,SOCK_STREAM,0);
  a.sin_family=AF_INET;
  a.sin_port=htons(502);
  a.sin_addr.s_addr=inet_addr(ip);
  connect(s,(sockaddr*)&a,sizeof(a));
  send(s,buf,len,0);closesocket(s);}
int main(){
  char buf[256];int len;
  char *ip="127.0.0.1";
  while(1){
    craft_read_coils(buf,&len,0,16);
    send_packet(ip,buf,len);
    for(int i=0;i<10;i++){
      craft_write_coil(buf,&len,i,1);
      send_packet(ip,buf,len);
      craft_write_coil(buf,&len,i,0);
      send_packet(ip,buf,len);}
    craft_read_registers(buf,&len,0,16);
    send_packet(ip,buf,len);
    for(int i=0;i<10;i++){
      craft_write_register(buf,&len,
        i,1234);
      send_packet(ip,buf,len);}
    Sleep(1000);} return 0;}
```