Modbus Protocol

Understanding the Most Common Industrial Protocol

OT Security Learning Series

Document 200 | January 2026

Contributors: Matthias Niedermaier

Created with AI assistance

LEARNING RESOURCE

Contents

1 Introduction **3**
1.1 Why Modbus Matters . . . 3

2 Protocol Overview **3**
2.1 Modbus Variants . . . 3
2.2 Architecture . . . 3
2.3 Data Model . . . 3

3 Function Codes **4**

4 Security Concerns **4**
4.1 Design Limitations . . . 4
4.2 Common Attack Vectors . . . 4
4.3 Attack Example . . . 5

5 Security Mitigations **5**
5.1 Network-Level Controls . . . 5
5.2 Monitoring and Detection . . . 5
5.3 Modbus/TCP Security Extension . . . 6

6 Further Reading **6**

1 Introduction

Modbus is one of the oldest and most widely deployed industrial communication protocols. Developed by Modicon in 1979, it has become the de facto standard for connecting industrial electronic devices due to its simplicity and openness.

Information

Despite being over 45 years old, Modbus remains ubiquitous in industrial environments. Understanding its operation and security limitations is essential for OT security professionals.

1.1 Why Modbus Matters

- **Ubiquity:** Found in virtually every industrial sector
- **Simplicity:** Easy to implement and troubleshoot
- **Legacy:** Many systems will run Modbus for decades to come
- **Attack surface:** No built-in security makes it an easy target

2 Protocol Overview

2.1 Modbus Variants

Variant	Medium	Description
Modbus RTU	Serial (RS-232/485)	Binary encoding, compact, most common serial variant
Modbus ASCII	Serial (RS-232/485)	ASCII encoding, human-readable, slower
Modbus TCP	Ethernet (TCP/IP)	RTU over TCP, port 502, most common today
Modbus UDP	Ethernet (UDP/IP)	Less common, no connection guarantee

2.2 Architecture

Modbus uses a master-slave (client-server) architecture:

Modbus Communication Model

- **Master/Client:** Initiates all communication (HMI, SCADA, PLC)
- **Slave/Server:** Responds to requests (sensors, actuators, I/O modules)
- **Unit ID:** Addresses slaves (1–247 for serial, 0–255 for TCP)
- **Request-Response:** Master sends request, slave responds

2.3 Data Model

Modbus defines four primary data types:

Type	Access	Size	Typical Use
Coils	Read/Write	1 bit	Digital outputs (relays, actuators)
Discrete Inputs	Read Only	1 bit	Digital inputs (switches, sensors)
Holding Registers	Read/Write	16 bits	Analog outputs, setpoints, config
Input Registers	Read Only	16 bits	Analog inputs (measurements)

3 Function Codes

Modbus operations are defined by function codes:

Code	Function	Description
0x01	Read Coils	Read status of digital outputs
0x02	Read Discrete Inputs	Read status of digital inputs
0x03	Read Holding Registers	Read analog/config values
0x04	Read Input Registers	Read analog input values
0x05	Write Single Coil	Set one digital output
0x06	Write Single Register	Set one register value
0x0F	Write Multiple Coils	Set multiple digital outputs
0x10	Write Multiple Registers	Set multiple register values

Warning

Function codes 0x05, 0x06, 0x0F, and 0x10 allow writing to devices. An attacker with network access can use these to manipulate physical processes without any authentication.

4 Security Concerns

4.1 Design Limitations

Critical

Modbus has no built-in security features:

- No authentication – any client can communicate with any server
- No encryption – all data transmitted in plaintext
- No integrity checking – packets can be modified in transit
- No authorization – no access control for function codes

4.2 Common Attack Vectors

1. **Reconnaissance:** Scanning for Modbus devices (port 502)
2. **Eavesdropping:** Capturing process data from network traffic
3. **Replay attacks:** Recording and replaying valid commands
4. **Command injection:** Sending unauthorized write commands
5. **Denial of service:** Flooding devices with requests

6. **Man-in-the-middle:** Intercepting and modifying communications

4.3 Attack Example

A simple Modbus TCP write command to set a coil:

```
# Transaction ID: 0x0001
# Protocol ID: 0x0000 (Modbus)
# Length: 0x0006
# Unit ID: 0x01
# Function: 0x05 (Write Single Coil)
# Address: 0x0000
# Value: 0xFF00 (ON)

00 01 00 00 00 06 01 05 00 00 FF 00
```

Tip

This 12-byte packet is all that's needed to turn on a coil. No credentials, no handshake, no verification. If you can reach the device, you can control it.

5 Security Mitigations

Since Modbus cannot be secured at the protocol level, defense must rely on compensating controls:

5.1 Network-Level Controls

- **Network segmentation:** Isolate Modbus devices in dedicated VLANs
- **Firewalls:** Restrict access to port 502 to authorized systems only
- **Deep packet inspection:** Use OT-aware firewalls to filter function codes
- **Encryption tunnels:** Wrap Modbus in VPN or TLS tunnels

5.2 Monitoring and Detection

- **Network monitoring:** Detect anomalous Modbus traffic patterns
- **Baseline behavior:** Alert on unexpected function codes or addresses
- **Rate limiting:** Detect flooding or scanning attempts
- **Logging:** Record all Modbus transactions for forensics

5.3 Modbus/TCP Security Extension

Information

The Modbus organization released a security specification that adds TLS encryption and role-based access control. However, adoption remains limited due to legacy device constraints.

6 Further Reading

Specifications

- **Modbus Organization** – Protocol Specifications
  https://www.modbus.org/modbus-specifications
- **Modbus/TCP Security** – Protocol Specification
  https://www.modbus.org/technical-resources

Security Resources

- **CISA** – ICS-CERT Advisories
  https://www.cisa.gov/news-events/ics-advisories
- **NIST SP 800-82** – Guide to OT Security
  https://csrc.nist.gov/pubs/sp/800/82/r3/final