Formal Verification for Discovering and Validating Cybersecurity Vulnerabilities in the Energy Sector

Matthew Boeding, Michael Hempel and Hamid Sharif

Department of Electrical and Computer Engineering, College of Engineering

Motivation

Operational Technology (OT) systems are becoming more **interconnected** and are **converging with IT** systems.

Cybersecurity priorities traditionally were different between OT and IT systems:

- Availability is paramount in OT systems
- Confidentiality is a higher priority in IT systems

Different protocol implementations can create vastly different behaviors to cyber attacks.

Protocol specifications must be manually evaluated by each vendor to create their own protocol implementation.

Challenges

Protocol Specifications are limited in scope:

- Reliant on industry experts to create robust device implementations
- Devices can have large discrepancies in handling a large amount of traffic
- May assume a "Defense in Depth" strategy, that leave devices vulnerable to protocol specific attacks

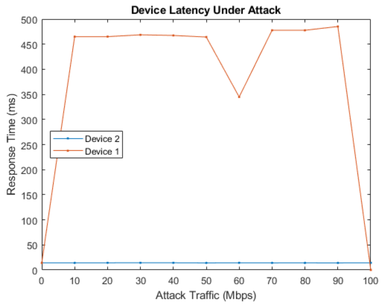


Figure 1. Effect of Cyber Attack on Device Response Time [2]

- A formalized method for identifying and verifying vulnerabilities would facilitate more robust devices

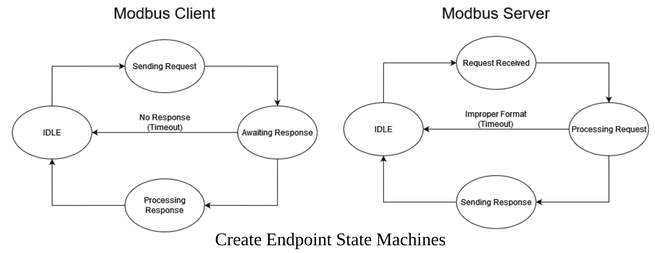
Contribution

Formal Verification Models of Protocol Specifications:

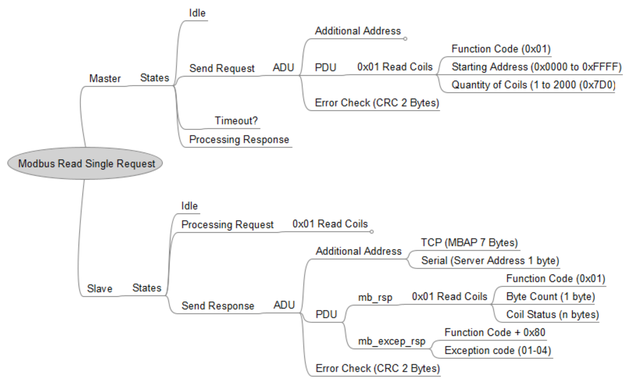
- Created using Construction and Analysis of Distributed Processes (CADP) [1]
- Utilizes Language of Temporal Ordering Specification (LOTOS) to generate a state machine model of Modbus Protocol Specification
- Simplify model to avoid an exponential growth of states by translating every field value into a binary decision
- Identify risks within specification and implementation cybersecurity, solely from protocol specifications
- Create a method of identifying vulnerabilities from protocol specifications

Methodology

1. **Manually Evaluate Protocol for Expected Behavior**



Create Endpoint State Machines



Identify Individual Packet Structure

Packet Field	Possible Packet Value →	Maps to Model Value
Server Address	0x00 – 0x7F	CORRECT, INCORRECT
Function Code	0x01	READ_COILS
Starting Address	0x0000 – 0xFFFF	VALID, INVALID
Value	0x0000 – 0x7D0	VALID, INVALID
CRC	0x0000 – 0xFFFF	CORRECT, INCORRECT
Possible Values	1.09e+15	16

Translate Packet Fields into Binary Decisions to Reduce State Space

2. **Expand Model to Encompass All Possible Packet Values**

Example: Modbus contains varying packet structure and processing steps

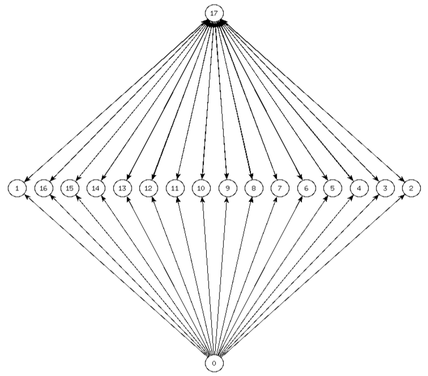
- Client:
  - 19 Request Function Codes
    - Can be reduced to 12 common packet formats
- Server:
  - 19 Response Function Codes
    - Can be reduced to 15 common packet formats
  - 19 Exception Function Codes
    - Can be reduced to 1 common packet format

Fields	Values
Function Code	19
Sub-Function Code	10
MEI Code	2
Server Address	2
Internal Address	2
Internal Value	2
Internal Quantity	2
Byte Count	2
CRC	2
Exception Error	4

Results

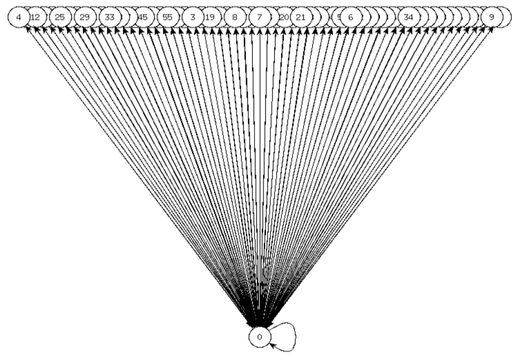
Single-Message Model:

Example: Read Coils Message has 16 possible packet value, resulting in 18 states



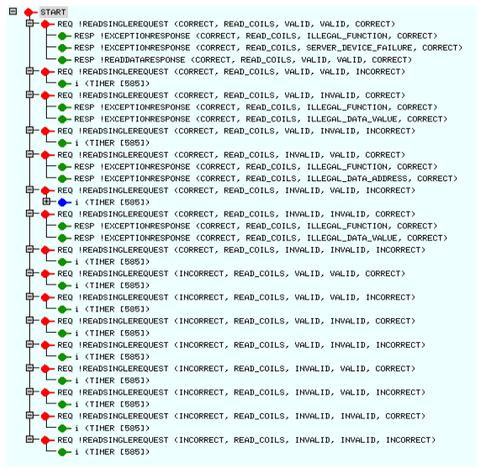
Full Protocol Model:

Represents all protocol messages; States reducible from **12,542** to **60** states; Able to reasonably view and explore model



Model Applications:

- Can be used to find vulnerabilities: transitions labelled with a hidden label ("i" in the image to the right)
- Can also be found directly from model walk, as shown below for a Read Coils message (for example with incorrect Server Address or CRC)
- Useful for targeted analysis of specific protocol operations or for holistic protocol review




```
!READSINGLEREQUEST (CORRECT, READ_COILS, VALID, VALID, INCORRECT)
!READSINGLEREQUEST (CORRECT, READ_COILS, VALID, INVALID, INCORRECT)
!READSINGLEREQUEST (CORRECT, READ_COILS, INVALID, VALID, INCORRECT)
!READSINGLEREQUEST (CORRECT, READ_COILS, INVALID, INVALID, INCORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, VALID, VALID, CORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, VALID, VALID, INCORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, VALID, INVALID, CORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, VALID, INVALID, INCORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, INVALID, VALID, CORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, INVALID, VALID, INCORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, INVALID, INVALID, CORRECT)
!READSINGLEREQUEST (INCORRECT, READ_COILS, INVALID, INVALID, INCORRECT)
```

```
./modbus_singlefull.bcg:
created by generator
        18 states
        92 transitions
        49 labels
        initial state: 0
        no deadlock state
        branching factor: average = 5.11, minimal = 1, maximal = 24
        12 transition(s) with a hidden label ("i")
        deterministic behavior for all labels
```

Project Collaborators

This work was supported by the Nebraska Public Power District through the Nebraska Center for Energy Sciences Research at the University of Nebraska-Lincoln.

NEBRASKA CENTER FOR ENERGY SCIENCES RESEARCH

[1] Garavel, H., Lang, F., Mateescu, R. et al. "CADP 2011: a toolbox for the construction and analysis of distributed processes", Intl. Journal on Software Tools for Technology Transfer, vol. 15, 2013. DOI: 10.1007/s10009-012-0244-z

[2] Boeding, M., Hempel, M., Sharif, H., Lopez Jr., J., Perumalla, K., "A Flexible OT Testbed for Evaluating On-Device Implementations of IEC-61850 GOOSE", Intl. Journal on Critical Infrastructure Protection, 2023, (under review)