Project Title: SCADA (Supervisory Control and Data Acquisition) Network Passive Reconnaissance Framework

---

Short Project Description:

Passive reconnaissance framework for SCADA/ICS networks. This framework identifies devices, decodes industrial protocols, maps communication flows, and detects anomalies without disrupting operations, enabling secure and risk-free network visibility.

Component	Role
Passive Traffic Collector	Sniffs network without active probing
Protocol Decoder	Decodes SCADA/ICS protocols (Modbus, DNP3, S7Comm, etc)
Device and Service Mapper	Identifies devices (PLC, HMI, RTU, etc) and services
Anomaly Detector (Optional)	Flags non-standard behaviors
Reporting Module	Produces SCADA network inventory and risk report

---

Component Details:

1. **Passive Traffic Collector**:
    - Sniffs network traffic using:
        - Wireshark/Tshark
        - Custom packet capture tools
        - Etc
    - No active scanning (risk-free).
2. **Protocol Decoder**:
    - Understands industrial protocols like:
        - Modbus TCP
        - DNP3
        - IEC 104
        - S7Comm (Siemens)
        - Etc
3. **Device and Service Mapper**:
    - Identifies devices:
        - PLCs
        - HMIs
        - RTUs
        - Etc
    - Maps out:
        - Address spaces
        - Register maps
4. **Anomaly Detector**:
    - Detects:

        - Unexpected commands
        - New unknown devices appearing
5. **Reporting Module**:
    - Summarizes:
        - Device inventories
        - Communication flows
        - Security risks
        - Etc

Overall System Flow:

- Input: Passive network monitoring
- Output: SCADA network map and basic risk report
- Focus: **Zero-impact reconnaissance**.

Internal Functioning of Each Module:

1. Passive Traffic Collector

- **How it works**:
    - **Sniffer** listens in **promiscuous mode**.
    - Tools (e.g.):
        - **Tshark** for continuous capture
        - **Scapy** or **PyShark** for live packet analysis
        - **Etc**
- **Capture focus**:
    - No packet injection
    - Only observation
    - Filters SCADA-specific protocols (e.g., port 502 for Modbus)

---

2. Protocol Decoder

- **Decoding logic**:
    - For **Modbus TCP**:
        - Decode Transaction ID, Protocol ID, Unit ID, Function Code, Data, etc.
        - Example: "Read Holding Registers" request
    - For **DNP3**:
        - Understand link layer, transport layer, application layer structure, etc.
        - Parse object headers (e.g., Binary Input, Analog Input, etc).

---

3. Device and Service Mapper

- **Mapping**:
  - Extracts:
    - Device addresses (e.g., Modbus Unit IDs, DNP3 addresses, etc)
    - Services (function codes used)
    - Device types guessed via fingerprinting
- **Graph construction**:
  - Builds communication graphs:
    - Who talks to whom?
    - How often?
    - What operations are common?

---

4. Anomaly Detector

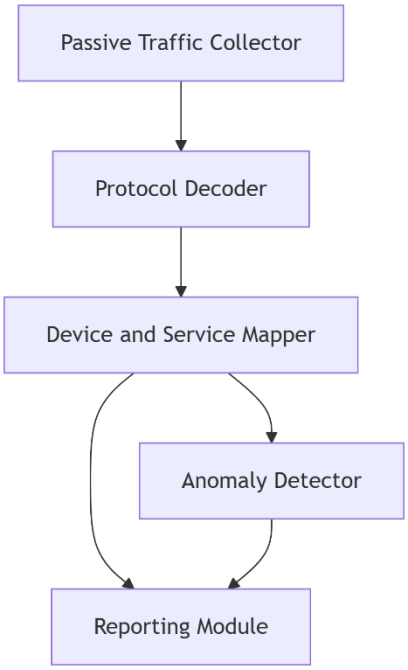
- **Behavior checks**:
  - New unknown devices appearing
  - Unexpected function codes (e.g., Writing when usually only Reading)
  - High volume Modbus Exception responses
  - Etc
- **Statistical thresholds** to avoid false positives.

---

5. Reporting Module

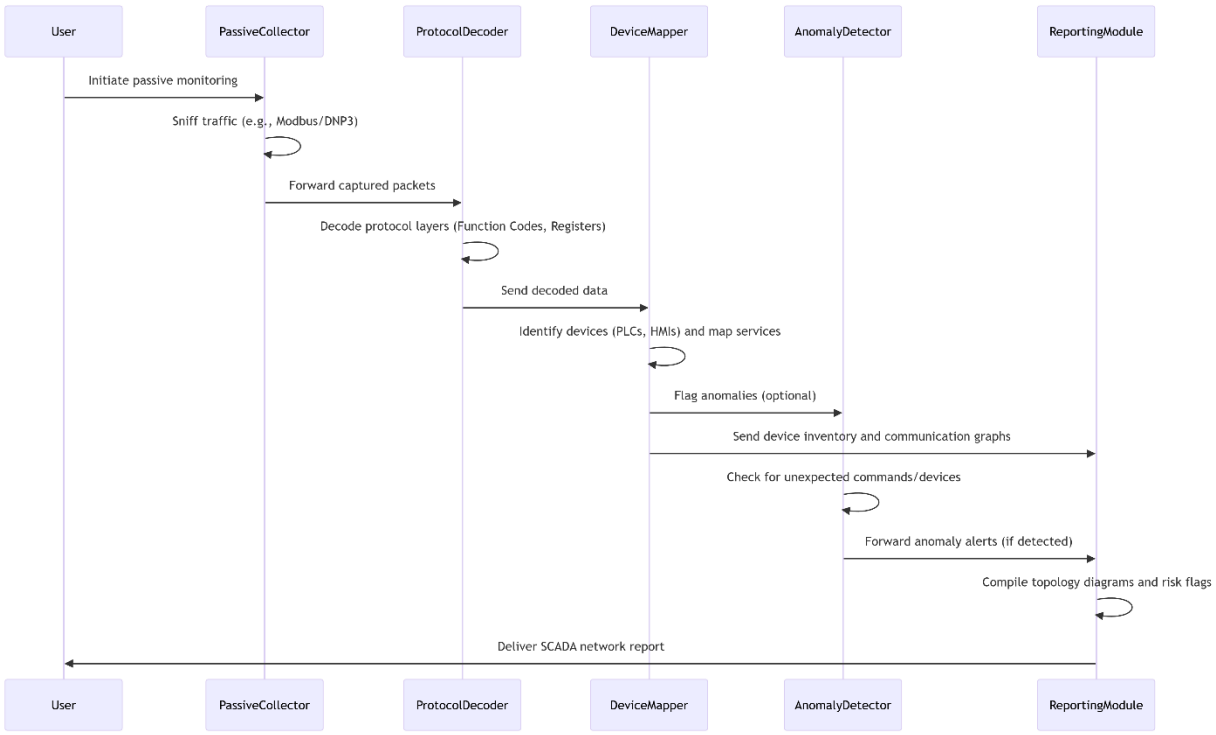
- **Organizes**:
  - Topology diagrams
  - Risk flags
  - Device tables (with IP, MAC, protocol details)
  - Etc

Component Diagram



- The **Passive Traffic Collector** feeds raw network traffic to the **Protocol Decoder**.
- The **Protocol Decoder** translates SCADA/ICS protocols (Modbus, DNP3, etc) into structured data for the **Device and Service Mapper**.
- The **Device and Service Mapper** identifies devices (PLCs, RTUs, etc) and maps communication flows.
- The optional **Anomaly Detector** flags unexpected behaviors (e.g., unauthorized commands, etc) and sends alerts to the **Reporting Module**.
- The **Reporting Module** aggregates all data into a risk report and topology map for the **User**.

Sequence Diagram



- The **User** triggers passive monitoring, and the **PassiveCollector** sniffs SCADA traffic without active probing.
- Captured packets are decoded by the **ProtocolDecoder** (e.g., Modbus function codes, DNP3 object headers, etc).
- The **DeviceMapper** identifies devices, services, and communication patterns, then sends this data to both the **AnomalyDetector** (for optional checks) and the **ReportingModule**.
- The **AnomalyDetector** analyzes for irregularities (e.g., new devices, abnormal commands, etc) and shares findings with the **ReportingModule**.
- The **ReportingModule** compiles a comprehensive report detailing the network topology, device inventory, and security risks for the **User**.

Detailed Project Description: SCADA Network Passive Reconnaissance Framework

A passive reconnaissance framework for SCADA/ICS networks. The framework identifies devices, decodes industrial protocols, maps communication flows, and detects anomalies without disrupting operations, enabling secure and risk-free network visibility.

---

1. System Overview

The framework passively monitors SCADA networks to inventory devices (PLCs, RTUs, HMIs, etc), decode industrial protocols (Modbus, DNP3, S7Comm, etc), and detect anomalies. It focuses on **zero-impact reconnaissance** to support asset management and security hardening.

---

2. Component Design & Implementation

2.1 Passive Traffic Collector

Functionality:

- Captures network traffic without active scanning to avoid operational disruption.

Implementation Steps (e.g.):

1. **Tool Selection**:
    - Use **Tshark** (CLI version of Wireshark) for continuous packet capture.
      ```
      tshark -i eth0 -f "tcp port 502 or udp port 20000" -w scada.pcap
      ```
    - **Scapy** or **PyShark** for real-time analysis in Python.
2. **Traffic Filtering**:
    - Focus on SCADA-specific ports:
        - Modbus TCP (502), DNP3 (20000), S7Comm (102), etc.
    - Exclude non-industrial traffic to reduce noise.
3. **Ethical Safeguards**:

- Ensure read-only access to network interfaces.
- Store captured data in encrypted formats (e.g., PCAPNG with AES).

Output:

- PCAP files containing SCADA protocol traffic.

Tools (e.g.):

- Tshark, Scapy, PyShark, etc.

2.2 Protocol Decoder

Functionality:

- Decodes SCADA/ICS protocols into structured data for analysis.

Implementation Steps (e.g.):

1. **Modbus TCP Decoding**:
   - Parse function codes (e.g., `03` = Read Holding Registers) and register addresses.
   - Use `pymodbus` library to extract transaction IDs, unit IDs, and payloads.
2. **DNP3 Decoding**:
   - Decode layers (link, transport, application) and object headers (e.g., analog inputs, etc).
   - Leverage `dnp3-python` for structured parsing.
3. **S7Comm Decoding**:
   - Extract job/function types (e.g., PLC stop/start commands).
   - Use custom parsers based on Siemens documentation.

Output:

- JSON-structured protocol data (e.g., `{"protocol": "Modbus", "function": "ReadHoldingRegisters", "address": 40001}`).

Tools (e.g.):

- `pymodbus`, `dnp3-python`, custom parsers, etc.

2.3 Device and Service Mapper

Functionality:

- Identifies devices and maps communication patterns.

Implementation Steps (e.g.):

1. **Device Identification**:
    - Correlate IP/MAC addresses with protocol-specific identifiers:
        - Modbus Unit IDs, DNP3 source addresses.
    - Fingerprint devices using response patterns (e.g., PLC model via S7Comm, etc).
2. **Communication Mapping**:
    - Build a graph using `NetworkX` to visualize:
        - **Nodes**: Devices (PLCs, HMIs, etc).
        - **Edges**: Frequency/type of interactions (e.g., read/write commands, etc).
3. **Service Enumeration**:
    - List supported function codes (e.g., Modbus `03` = Read, `06` = Write, etc).

Output:

- Network topology graph and device inventory (CSV/JSON).

Tools (e.g.):

- NetworkX, Pandas, SQLite, etc.

2.4 Anomaly Detector

Functionality:

- Flags deviations from baseline behavior (e.g., unauthorized commands).

Implementation Steps (e.g.):

1. **Baseline Establishment**:
    - Compute normal traffic patterns (e.g., average reads/hour, common function codes, etc).
2. **Rule-Based Detection**:
    - Define YAML rules
    - Example:

```
rules:
  - name: "Unexpected Write Command"
    condition: "protocol == 'Modbus' and function_code == '06' and h
our > 18:00"
    severity: "High"
```

3. **Statistical Analysis**:
    - Use Z-scores to detect traffic spikes or rare function codes.

Output:

- Alerts for anomalies (e.g., new devices, unexpected writes).

Tools (e.g.):

- YAML, Pandas, Scikit-learn, etc.

2.5 Reporting Module

Functionality:

- Generates reports for asset management and risk assessment.

Implementation Steps (e.g.):

1. **Report Content**:
    - **Device Inventory**: IP, MAC, protocol, function codes, etc.
    - **Topology Diagrams**: Use Graphviz or D3.js for interactive visualizations.
    - **Risk Assessment**: Highlight devices with write permissions or weak protocols (e.g., Modbus without TLS, etc).
2. **Formats**:
    - **HTML**: Jinja2 templates with filters and search.
    - **PDF**: Convert HTML using WeasyPrint.
    - **STIX/TAXII**: Export threat intelligence for SIEM integration.
    - **Etc**

Output:

- Professional reports for IT/OT teams.

Tools (e.g.):

- Jinja2, Graphviz, WeasyPrint, STIX/TAXII, etc.

---

3. Technology Stack (e.g.)

- **Packet Capture**: Tshark, Scapy, etc.
- **Protocol Decoding**: pymodbus, dn3-python, etc.
- **Mapping**: NetworkX, SQLite, etc.
- **Reporting**: Jinja2, Graphviz, STIX/TAXII, etc.

---

4. Evaluation & Validation

1. **Accuracy Testing**:
    - Deploy on a testbed with known devices (e.g., Siemens S7-1200 PLC, Schneider Electric HMIs, etc).
    - Verify device identification and protocol decoding accuracy.

2. **Anomaly Detection**:
    - Inject synthetic attacks (e.g., unauthorized write commands, etc) and measure detection rates.
3. **Performance**:
    - Measure packet capture throughput (packets/sec) and processing latency.

---

5. Development Roadmap

1. **Phase 1**: Implement Passive Traffic Collector and Protocol Decoder.
2. **Phase 2**: Build Device/Service Mapper and basic reporting.
3. **Phase 3**: Add Anomaly Detector and STIX/TAXII integration.
4. **Phase 4 (optional)**: Validate on operational SCADA networks (with authorization).

---

6. Challenges & Mitigations (optional)

- **Protocol Diversity**: Prioritize common protocols (Modbus, DNP3) first; extend later.
- **Encrypted Traffic**: Partner with vendors to analyze TLS-wrapped traffic (e.g., Modbus Secure).
- **Legacy Systems**: Use protocol simulation tools (e.g., Modbus Slave) for testing.

---

7. Glossary

- **PLC**: Programmable Logic Controller
- **RTU**: Remote Terminal Unit
- **HMI**: Human-Machine Interface
- **STIX/TAXII**: Standards for threat intelligence sharing
- **DNP3**: stands for Distributed Network Protocol version 3
