Project Title: ICS (Industrial Control Systems) Threat Simulation and Response Testing

---

Short Project Description:

An ICS threat simulation and response testing system. The goal is to create a cyber-resilient infrastructure by simulating attacks, testing detection capabilities, orchestrating responses, and generating actionable reports.

Component	Role
ICS Attack Simulation Engine	Simulates known ICS-specific attacks
Target System Emulator	Creates safe ICS device environments for testing
Detection Capability Tester	Measures if defenses detect simulated attacks
Response Orchestration Module	Triggers simulated incident response activities
Reporting Module	Provides threat detection and response gap reports

Component Details:

1. **ICS Attack Simulation Engine**:
    - Simulates attacks like:
        - Unauthorized Modbus writes
        - Replay attacks
        - Command injections
        - Etc
    - Based on real-world TTPs (e.g., MITRE ATT&CK for ICS, etc).
2. **Target System Emulator**:
    - Emulates devices:
        - Simulated PLCs
        - Simulated RTUs
        - Allows safe attack simulations without harming real systems.
3. **Detection Capability Tester**:
    - Tests if:
        - Intrusion Detection Systems (IDS)
        - Security Information and Event Management (SIEM)
    - Can detect attack simulations.
4. **Response Orchestration Module**:
    - Launches simulated incident responses:
        - Alerts
        - Operator notifications
        - Automated shutdowns (in emulation)
5. **Reporting Module**:
    - Details:

        - Which attacks were detected
        - Which were missed
        - Recommendations for improved defense
        - Etc

Overall System Flow:

- Input: ICS/SCADA simulation environment
- Output: Detection and response gap report
- Focus: **Testing and training for cyber-resilient critical infrastructure**.

Internal Functioning of Each Module:

1. ICS Attack Simulation Engine

- **Attack simulations**:
    - Unauthorized Modbus Write:
        - Force changing a register value (e.g., opening a valve, disabling alarms, etc).
    - Replay attacks:
        - Capture legitimate traffic, replay later to cause unwanted commands.
    - Session hijacks:
        - Insert spoofed packets mid-session.
    - Etc.
- **Tactics**:
    - Replaying old valid messages to cause state changes.
    - Crafting fake control messages.

---

2. Target System Emulator

- **Emulation**:
    - Simulate real PLC/RTU behavior.
    - Open source tools like:
        - **MBLogic** (Modbus server simulation)
        - **Conpot** (ICS honeypot emulator)
        - Etc
- **Safe environment**:
    - No physical PLC damage risk.
    - Full attack scenarios possible.

---

3. Detection Capability Tester

- **Validation**:
  - Send attack traffic ➔ watch if:
    - IDS/IPS systems detect anomalous patterns.
    - SIEMs generate alerts.
- **Techniques**:
  - Monitor system logs (Sysmon for Windows, auditd for Linux, etc).
  - Parse network flow logs.

---

4. Response Orchestration Module

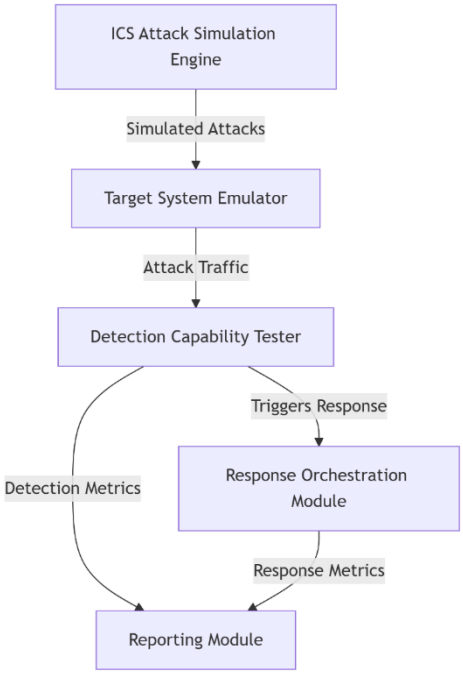
- **Simulated response actions**:
  - Simulate human operator reaction (phone calls, escalation, etc).
  - Test automatic containment actions (e.g., VLAN isolation, etc).

---

5. Reporting Module

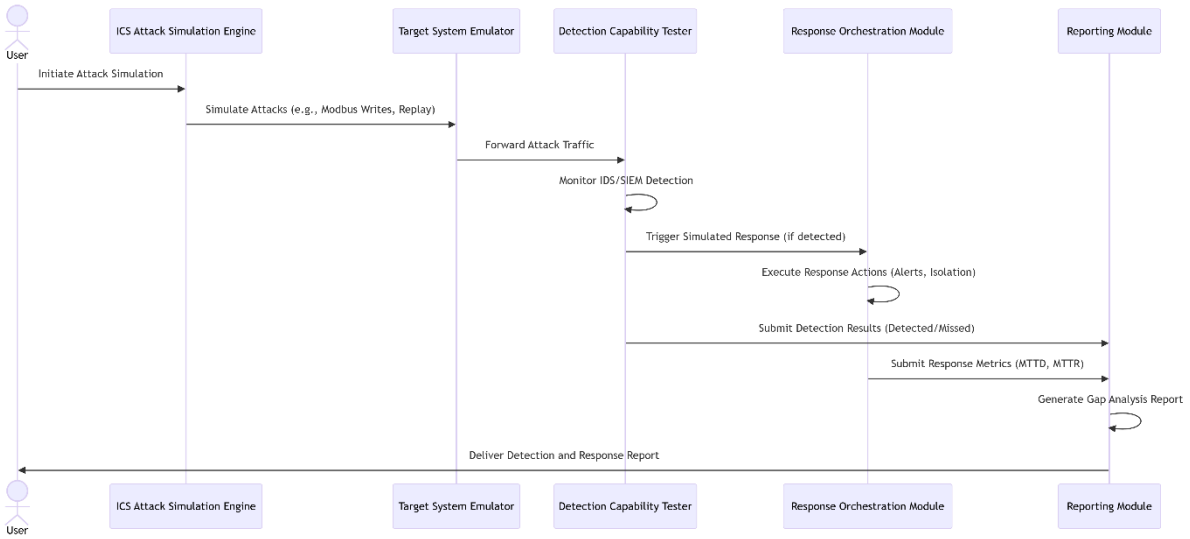
- **Report content**:
  - Which attacks succeeded.
  - Which were detected.
  - Mean time to detect (MTTD).
  - Mean time to respond (MTTR).
  - Recommendations (e.g., deploy better network segmentation, IDS signatures).
  - Etc.

Component Diagram



- **ICS Attack Simulation Engine** generates attacks (e.g., unauthorized Modbus writes, etc) and sends them to the **Target System Emulator**.
- The **Target System Emulator** (simulated PLCs/RTUs) forwards attack traffic to the **Detection Capability Tester**.
- The **Detection Capability Tester** evaluates defenses (IDS/SIEM) and triggers the **Response Orchestration Module** for incident response.
- Both **Detection Tester** (detection metrics) and **Response Module** (response metrics) feed data into the **Reporting Module**, which compiles the final report.

Sequence Diagram



- **User** starts the process by triggering the **Attack Simulation Engine**.
- Attacks are simulated on the **Target System Emulator**, which relays traffic to the **Detection Tester**.
- **Detection Tester** monitors defenses and, if attacks are detected, activates the **Response Orchestration Module** (e.g., alerts, automated shutdowns).
- Results from detection and response are aggregated by the **Reporting Module** into a gap analysis report for the user.

Detailed Project Description: ICS Threat Simulation and Response Testing

An ICS threat simulation and response testing system. The goal is to create a cyber-resilient infrastructure by simulating attacks, testing detection capabilities, orchestrating responses, and generating actionable reports.

---

1. System Components and Roles

1.1 ICS Attack Simulation Engine

Purpose: Simulate ICS-specific attacks using real-world Tactics, Techniques, and Procedures (TTPs).
Implementation Details (e.g.):

- **Attack Types**:
    - **Unauthorized Modbus Writes**: Modify register values (e.g., valve states, alarm thresholds, etc) using tools like `python-scapy` or custom scripts.
    - **Replay Attacks**: Capture legitimate Modbus/TCP traffic with `Wireshark` or `tcpdump`, then replay using tools like `tcpreplay`.
    - **Command Injection**: Inject malicious payloads into ICS protocols (e.g., CIP, DNP3 etc) using frameworks like `Metasploit` (ICS modules).
    - **Etc**.
- **TTPs**: Align with MITRE ATT&CK for ICS (e.g., "Command-Line Interface," "Man-in-the-Middle").
- **Tools**:
    - `Cobalt Strike` (for advanced attack simulations), Metasploit.
    - Custom Python scripts to generate attack traffic.
    - Etc

1.2 Target System Emulator

Purpose: Emulate ICS devices (PLCs, RTUs) in a safe environment.
Implementation Details (e.g.):

- **Emulated Devices**:
    - Simulate PLCs using `MBLogic` (Modbus server) or `OpenPLC`.
    - Emulate RTUs with `Conpot` (ICS honeypot).
- **Setup**:
    - Deploy emulators on virtual machines (VMware, VirtualBox) or Docker containers.
    - Configure network isolation (e.g., VLANs) to prevent accidental impact on real systems.
- **Safety**: Ensure no physical hardware is connected; use software-based I/O (e.g., simulated sensors/actuators).

1.3 Detection Capability Tester

Purpose: Validate if IDS/IPS and SIEM systems detect attacks.
Implementation Details (e.g.):

- **Detection Tools**:
    - **Network IDS**: Deploy `Suricata` or `Zeek` with ICS-specific rules (e.g., `Digital Bond's ICS Snort Rules`).
    - **Host IDS**: Use `Sysmon` (Windows) or `auditd` (Linux) for log collection.
    - **SIEM**: Configure `Elastic SIEM` or `Splunk` to aggregate and analyze logs.
- **Validation Process**:
    1. Run attack simulations.
    2. Monitor SIEM dashboards for alerts (e.g., "Unauthorized Modbus Write detected").
    3. Analyze network traffic for anomalies (e.g., unexpected command sequences).

1.4 Response Orchestration Module

Purpose: Simulate incident response actions.
Implementation Details (e.g.):

- **Automated Responses**:
    - Trigger VLAN isolation via scripts (e.g., `Python + Cisco API`).

    - Send automated alerts to Slack/email using `Zapier` or `PagerDuty`.
- **Human-in-the-Loop**:
    - Simulate operator actions (e.g., manual shutdown via HMI emulators like `ScadaBR`).
- **Tools**:
    - Use SOAR platforms like `TheHive` or `Shuffle` for playbook automation.

1.5 Reporting Module

Purpose: Generate detection and response gap analysis.
Implementation Details (e.g.):

- **Metrics**:
    - Mean Time to Detect (MTTD).
    - Mean Time to Respond (MTTR).
    - Attack success/failure rates.
    - Etc
- **Tools**:
    - Use `Jupyter Notebooks` for data analysis and visualization (e.g., `Matplotlib`, `Pandas, etc`).
    - Generate PDF reports with `LaTeX` or `Python ReportLab`.
- **Recommendations**: Include actionable steps (e.g., "Update IDS rules for Modbus Function Code 6").

---

2. System Integration and Component Interaction

1. **Attack Simulation**:
    - The **ICS Attack Simulation Engine** sends attack traffic (e.g., unauthorized Modbus writes, etc) to the **Target System Emulator**.
    - Example command: `python simulate_modbus_write.py --target 192.168.1.10 --register 40001 --value 1`.
2. **Traffic Relay**:

    - The **Target System Emulator** forwards attack traffic to the **Detection Capability Tester**.
3. **Detection Testing**:
    - The **Detection Tester** evaluates IDS/SIEM logs and triggers the **Response Orchestration Module** if attacks are detected.
4. **Response Execution**:
    - The **Response Module** executes actions (e.g., VLAN isolation, alerts, etc).
5. **Reporting**:
    - Data from detection and response modules is aggregated into the **Reporting Module** for analysis.

---

3. General Implementation Steps

3.1 Environment Setup (e.g.)

- **Hardware**: Use a dedicated server or high-performance PC (16GB RAM, 4-core CPU).
- **Software**:
    - Install Ubuntu 22.04 LTS as the base OS.
    - Use Docker for containerized emulators (e.g., `docker run -d --name conpot conpot/conpot`).
- **Network Configuration**:
    - Isolate the lab network (e.g., 192.168.100.0/24).
    - Configure firewall rules to restrict external access.

3.2 Attack Simulation Setup (e.g.)

- Clone the ICS attack repository:
  ```
  git clone https://github.com/ics-attack-simulator/scripts
  ```
- Configure attack parameters in `config.yaml` (e.g., target IP, attack type).

3.3 Detection System Configuration (e.g.)

- Install Suricata with ICS rules:

```
sudo apt install suricata
wget https://ics-rules.com/suricata.rules -O /etc/suricata/rules/
```

- Configure Elastic SIEM to ingest Suricata alerts.

3.4 Response Playbook Example (e.g.)

- **Playbook**:
    1. If "Unauthorized Modbus Write" is detected:
        - Send alert to SOC.
        - Isolate the compromised VLAN.
        - Log incident in the SIEM.

3.5 Validation and Testing (e.g.)

- **Test 1**: Simulate a replay attack. Verify if Suricata generates an alert.
- **Test 2**: Measure MTTD for command injection attacks.

---

4. Evaluation Criteria

1. **Detection Accuracy**: Percentage of attacks detected by IDS/SIEM.
2. **Response Effectiveness**: Time taken to contain attacks (MTTR).
3. **Report Quality**: Clarity of recommendations (e.g., "Deploy network segmentation between OT and IT").

---

5. Safety and Ethics

- **Safety**: Never connect emulated systems to real ICS hardware.
- **Ethics**: Use this system only in authorized environments.

---

6. Tools and Resources (e.g.)

- **Attack Simulation**: Metasploit, Cobalt Strike, custom Python scripts, etc.
- **Emulation**: Conpot, MBLogic, OpenPLC, etc.
- **Detection**: Suricata, Zeek, Elastic SIEM, etc.
- **Response**: TheHive, Shuffle, Cisco API, etc.
