# Detecting Intruders in OT Networks

Mehdi Sayyad
Ali Mirzaee

Aban 1404

# Who We Are

**Mehdi Sayad**

Security Architect/Consultant

Over 15 years working different cybersecurity projects in IT/OT

Interested fields:

- Penetration testing and risk assessment
- Network security architect
- Attack detection and analyses
- Software security
- ICS OT Network security

**Ali Mirzaee**

DevSecOps Engineer

+5 years in IT/infra Automation

Interested fields:

- CI/CD and secure pipelines
- Security automation
- Infrastructure as Code
- Ias security
- Attack detection and intrusion analyze

# Agenda

OT elements Fundamentals
IT vs OT security differences
ICS attack case studies (Stuxnet, TRITON, Colonial Pipeline)
Threat actors targeting OT
Common Vulns & Attack Vectors
Kill chain & TTP Analysis in OT
OT/ICS Network Visibility & Intrusion Detection
Hands-on tools and exercise

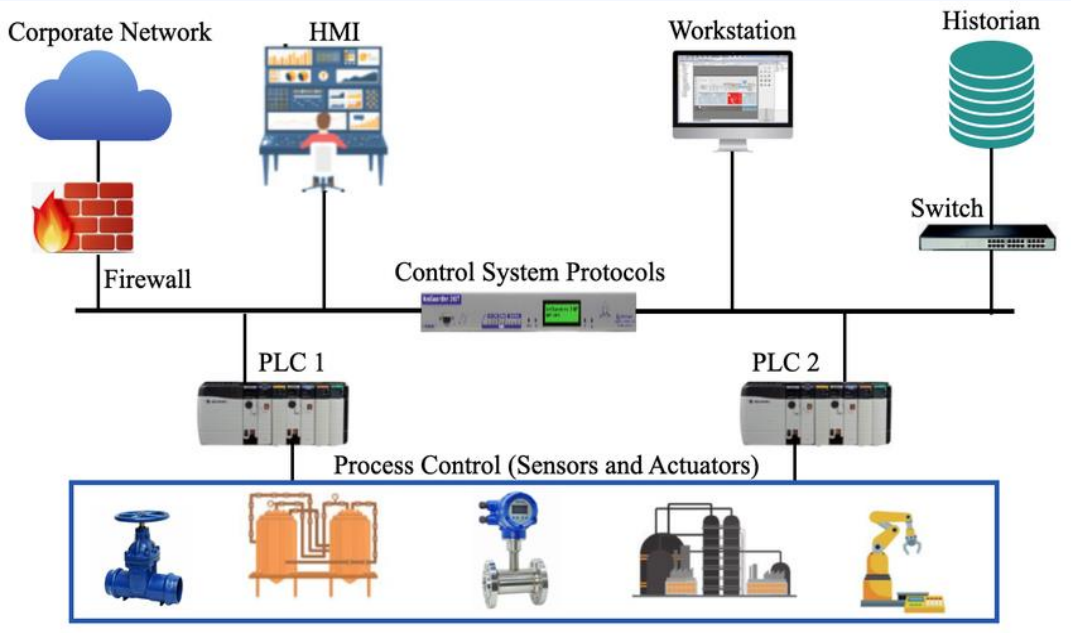
# 01

# OT/ICS Environment and Threat Landscape

Understanding OT/ICS Networks and Attack vectors

# Intro to OT Environments

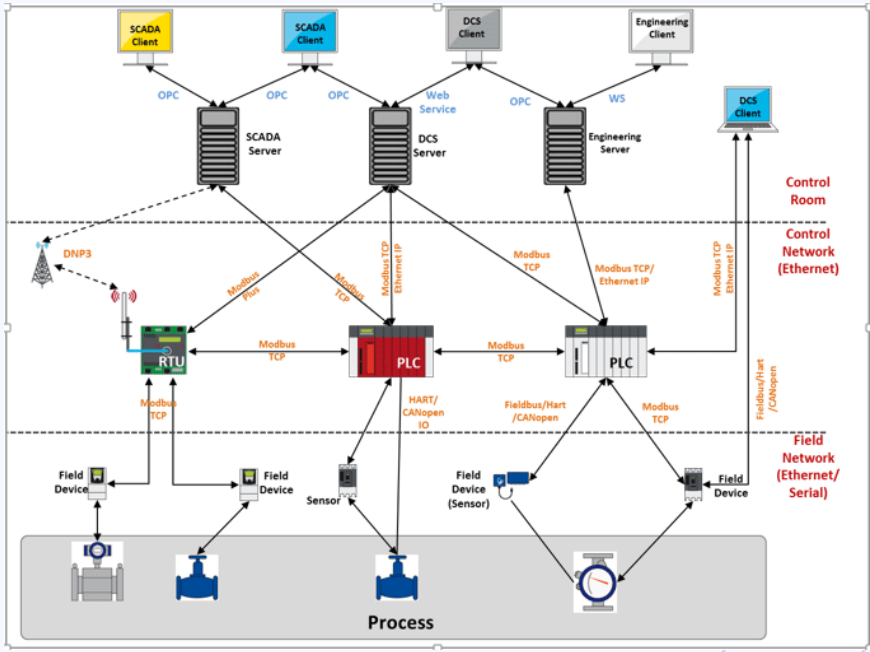
- SCADA
- HMI
- EWS
- PLC
- RTU
- IED
- SIS
- Communication devices
- …..
- …



# Typical OT Network: Protocols

Fieldbus/serial:
Fieldbus: Modbus RTU, Profibus, DeviceNET, CANOpen
IndustrialEthernet:
Modbus TCP , Profinet, Ethernet/IP, Ethernet CAT

Wireless: 802.15.4, 6LoWPAN, Bluetooth/L, Cellular LORA Wi-Fi WirelessHART, ZigBee




OT network zones: Enterprise → DMZ → Control → Safety

# IT vs OT Security

**IT Goal:** Protect data (Confidentiality, Integrity, Availability).

**OT Goal:** Protect human safety and physical processes (Availability, Integrity, Confidentiality).

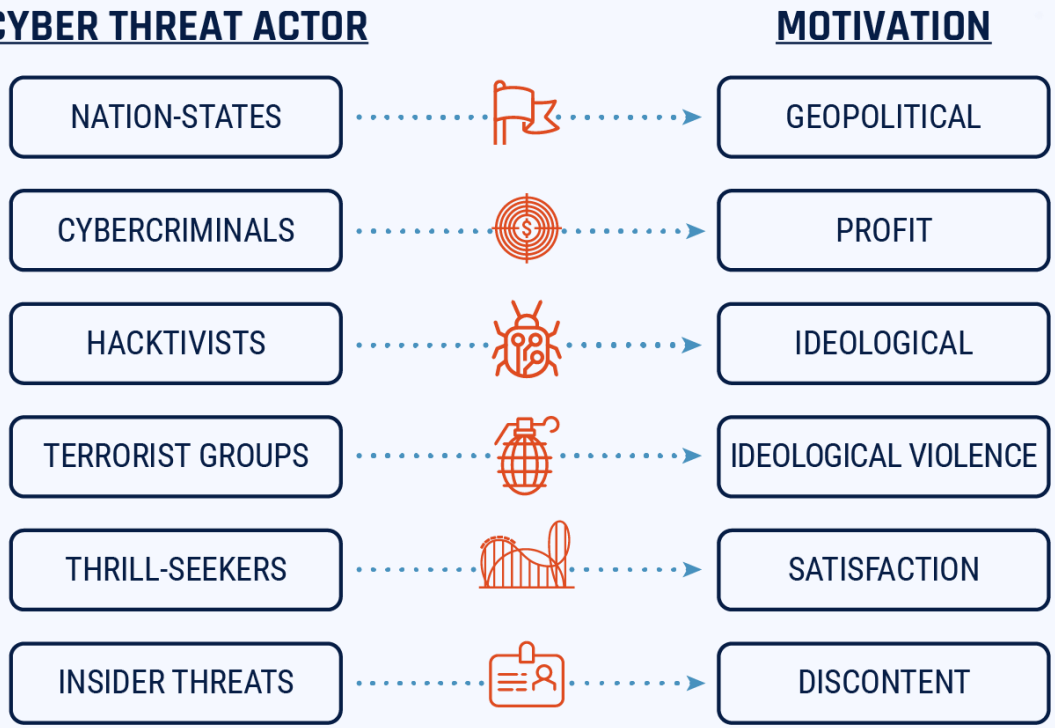
# IT vs OT Security

| Aspect | IT Security | OT Security |
| --- | --- | --- |
| **Focus** | Data protection and business continuity | Physical safety and process uptime |
| **Environment** | Office/digital networks | Industrial/legacy systems |
| **Downtime Tolerance** | Can handle updates/restarts | Avoids changes to prevent disruptions |
| **Threats** | Phishing, ransomware on data | Malware targeting hardware/control |
| **Update Cycle** | Frequent patching | Rare, to maintain stability |

# Common weakness in OT/ICS

- Lack of network segmentation allowing lateral movement from IT to OT.
- Weak or default credentials, poor identity and access management.
- Legacy and unpatched systems with known vulnerabilities.
- Unsecured remote access and poor monitoring.
- Synchronization of IT and OT identity services leading to wider exposure
- Limited logging and anomaly detection preventing early attack detection

# Threat Actors Targeting OT

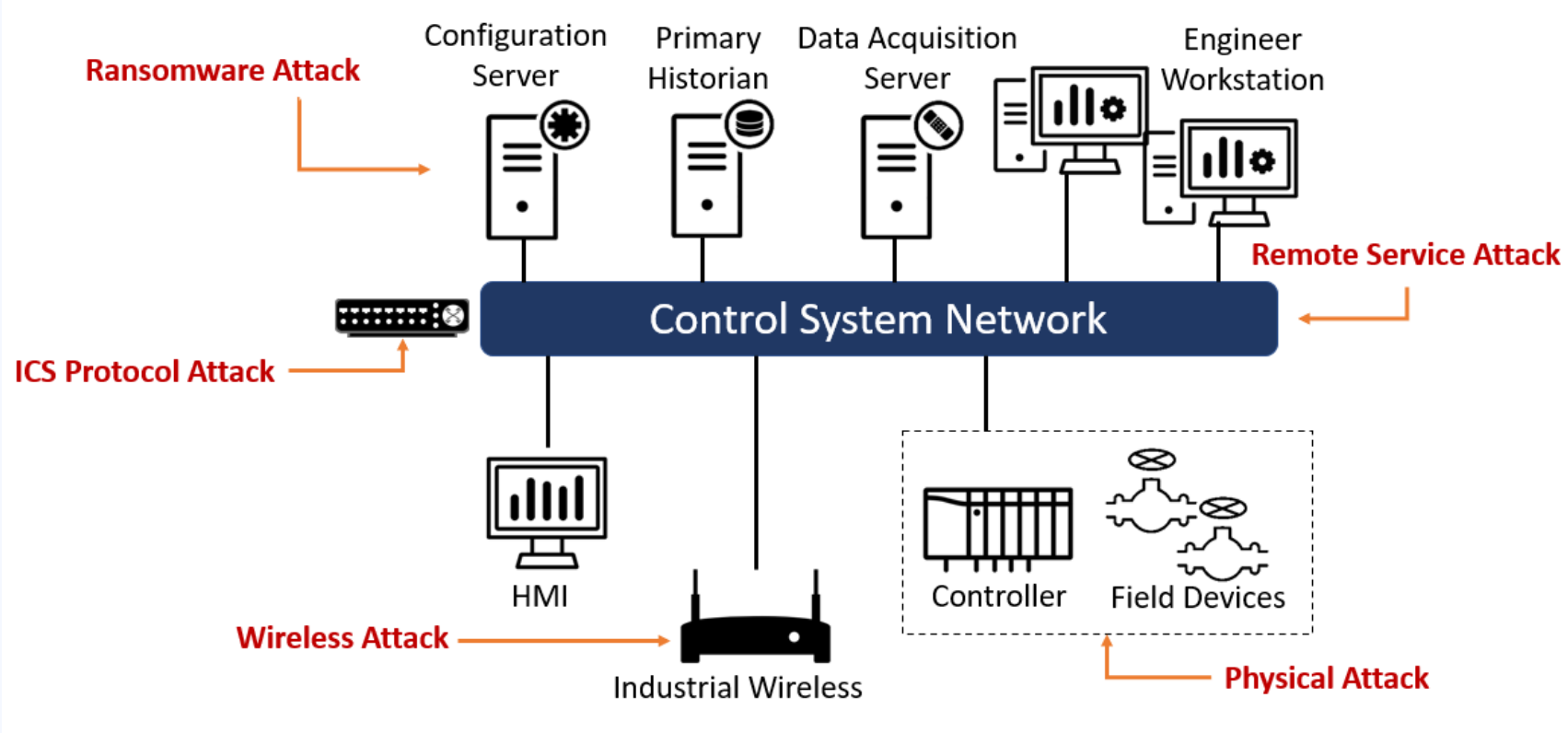


# OT Attack Vectors

How attackers infiltrate OT/ICS environments?

- **Phishing and Social Engineering:** Deceptive emails or communications trick employees into revealing credentials or installing malware, exploiting human vulnerabilities for initial access.
- **Compromised Remote Access:** Weak controls like default credentials or unsecured VPNs/RDP allow direct entry via vendor or support systems.
- **Removable Media (USB Drives) :** Infected devices introduce malware, bypassing air-gaps through physical access and lax controls.
- **Exploitation of OT Protocol Weaknesses:** Insecure protocols (e.g., Modbus, DNP3) without encryption enable command spoofing and lateral movement.
- **Vulnerabilities in Legacy Systems:** Unpatched, outdated OT hardware/software provide exploitable flaws for easy breaches.
- **Supply Chain and Software Compromise:** Malicious code inserted via updates or hardware from trusted suppliers infiltrates networks.

# OT Attack Vectors

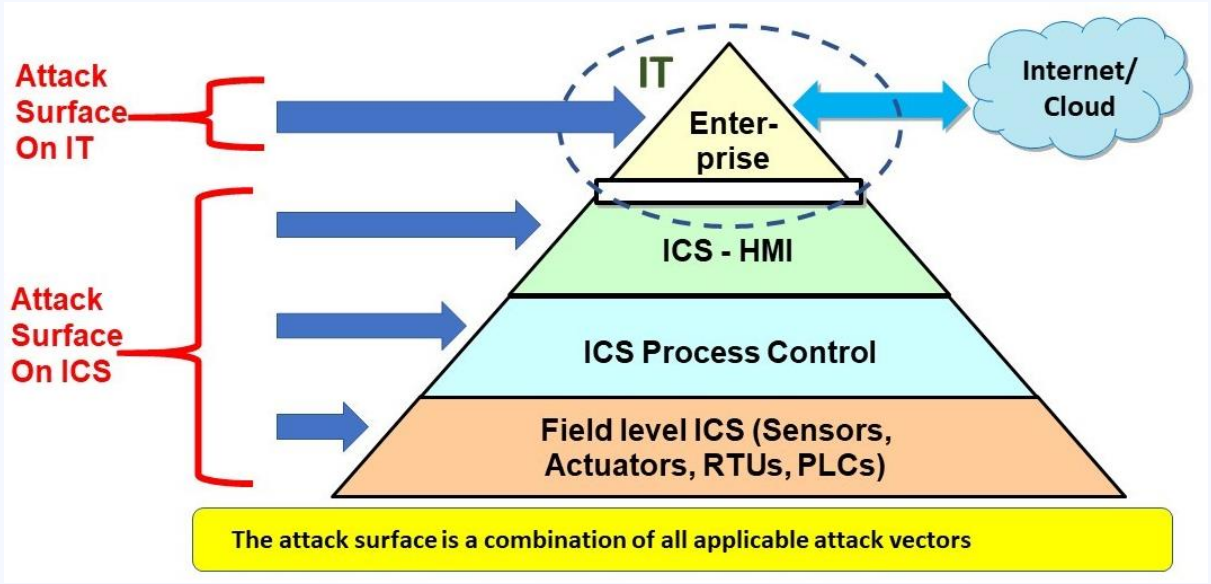


# Real-world attacks

| Date | Attack Name | Target | Description |
|---|---|---|---|
| 2010 | Stuxnet | Iranian nuclear centrifuges | First known malware targeting OT; disrupted uranium enrichment by manipulating Siemens PLCs via USB infection. |
| 2016 | Industroyer | Ukraine power grid | Malware causing blackout by controlling circuit breakers and relays through OT protocols. |
| 2017 | Triton | Saudi petrochemical plant SIS | Targeted safety instrumented systems (SIS) to disable safety controls, risking catastrophic failures. |
| 2021 | Oldsmar Water Plant | Water treatment facility, USA | Attempted to poison water supply by altering chemical levels through stolen remote credentials. |
| 2021 | Colonial Pipeline | US fuel pipeline system | Ransomware attack caused shutdown, disrupting fuel supply nationwide. |
| 2022 | Industroyer2 | Ukraine power grid (thwarted) | Updated malware variant caught before causing the prior damage level. |
| 2022 | Toyota Supplier Attack | Kojima Industries (Toyota supply) | Ransomware disrupted production management systems, halting plants globally. |

# Why Network Visibility Matters in ICS

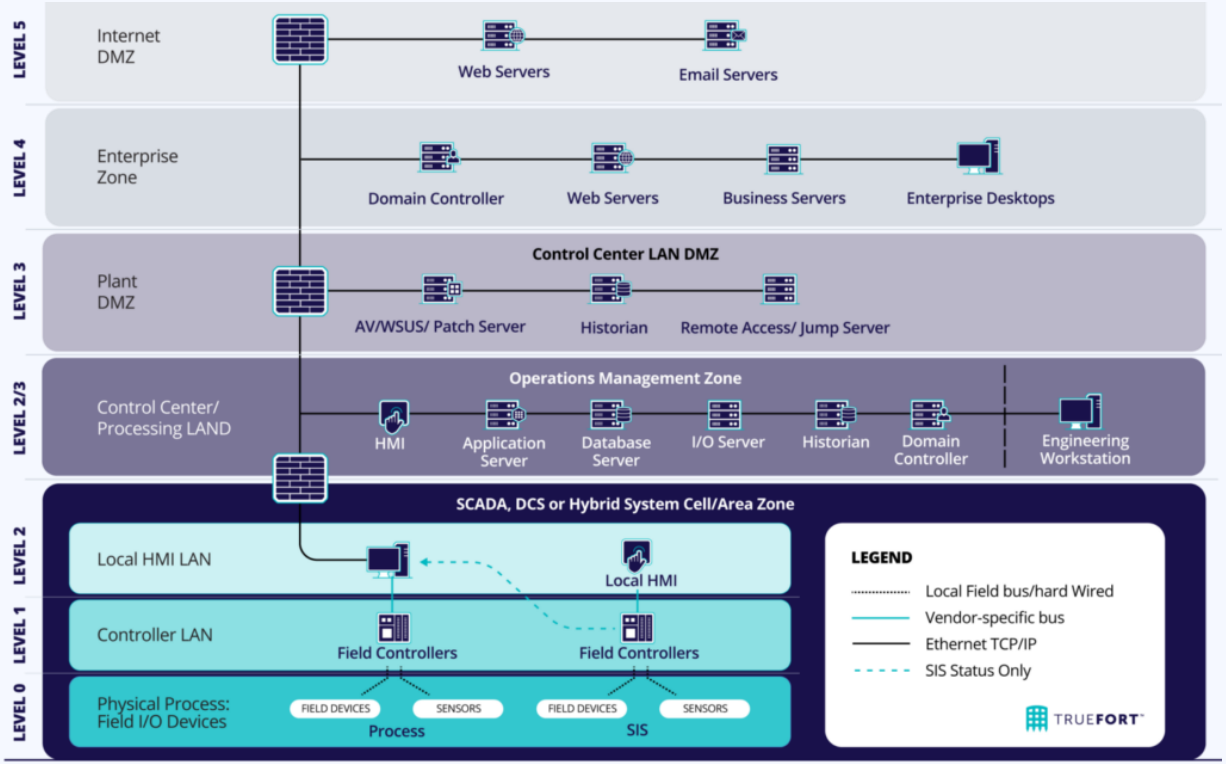
ICS Attacks Are Multi-Stage




Total View for OT Threats

Without visibility we cant catch intruders and defense advances threats

# Purdue Model for OT



# Analysis of attacks

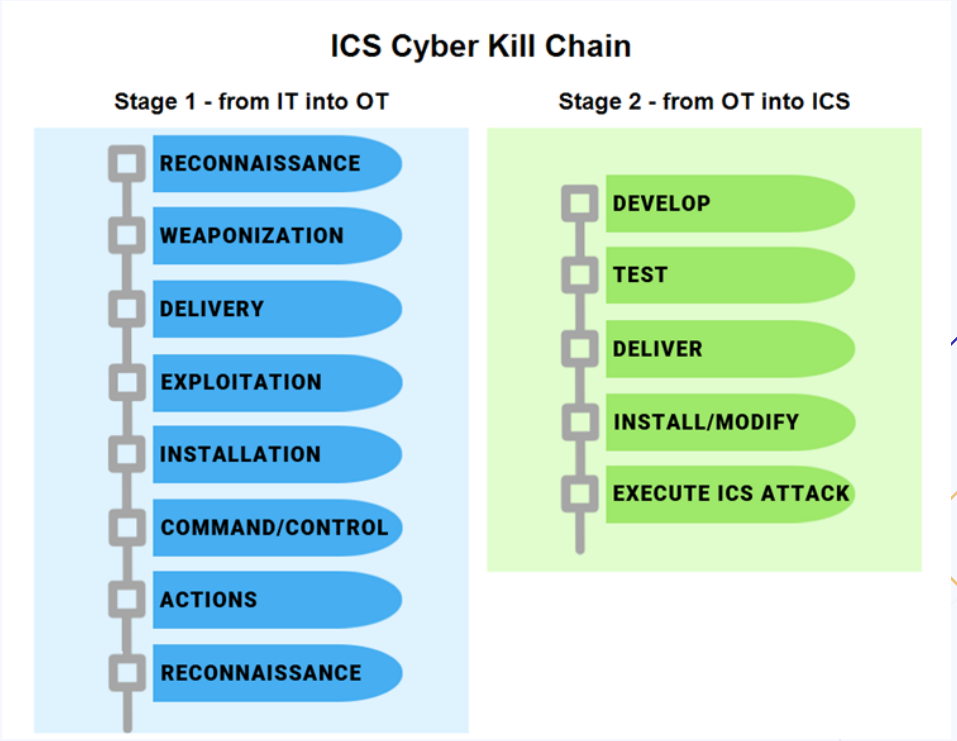
Attacks on OT and ICS systems are modeled by the ICS Cyber Kill Chain and the MITER ATT&CK for ICS Matrix

# ICS kill chain

The ICS Cyber Kill Chain is an adaptation of Lockheed Martin's original Cyber Kill Chain framework, extended for OT/ICS environments. It divides attacks into two stages:

**Stage 1 (IT-Focused Intrusion)** for gaining initial access and persistence in enterprise networks

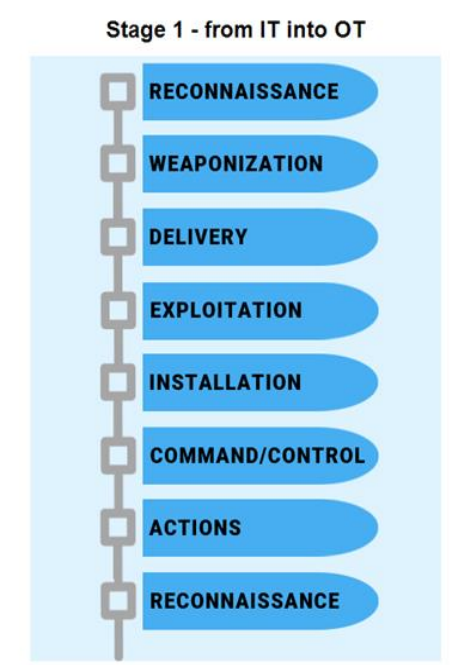
**Stage 2 (OT-Focused Attack)** for targeting ICS-specific components.



# ICS kill chain

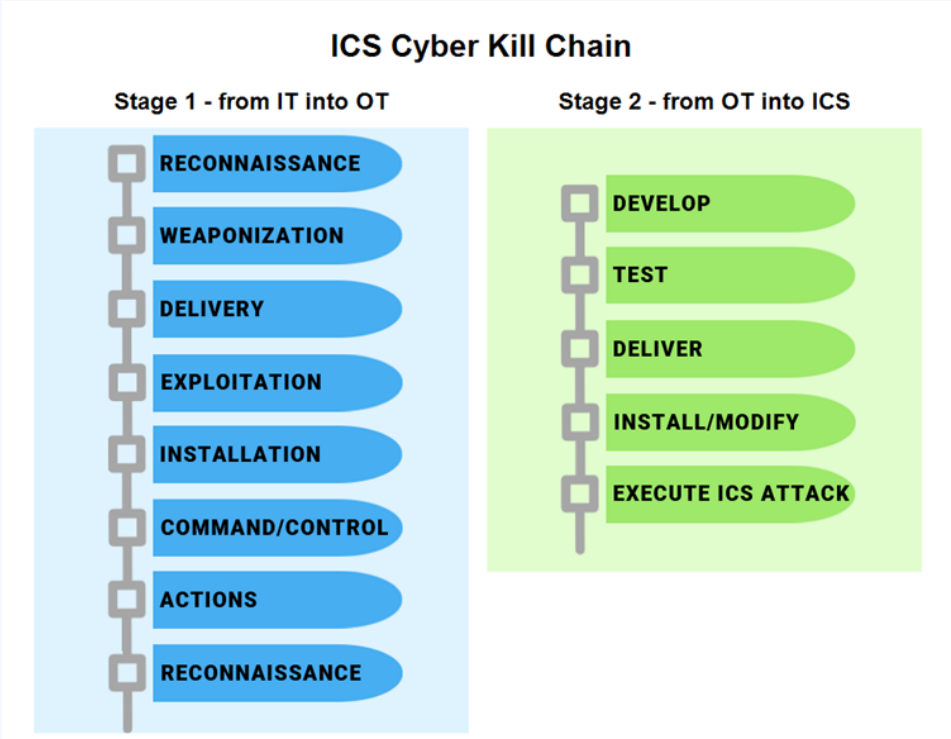
**Stage 1**:

- **Planning**: Reconnaissance using OSINT, vulnerability scanning, and target profiling.
- **Preparation**: Weaponization of tools/malware and selection of delivery methods.
- **Cyber Intrusion**: Delivery and exploitation for initial access.
- **Management and Enablement**: Establishing C2 channels and basic persistence.
- **Sustainment, Entrenchment, Development, and Execution**: Lateral movement, data exfiltration, and preparation for OT pivot.

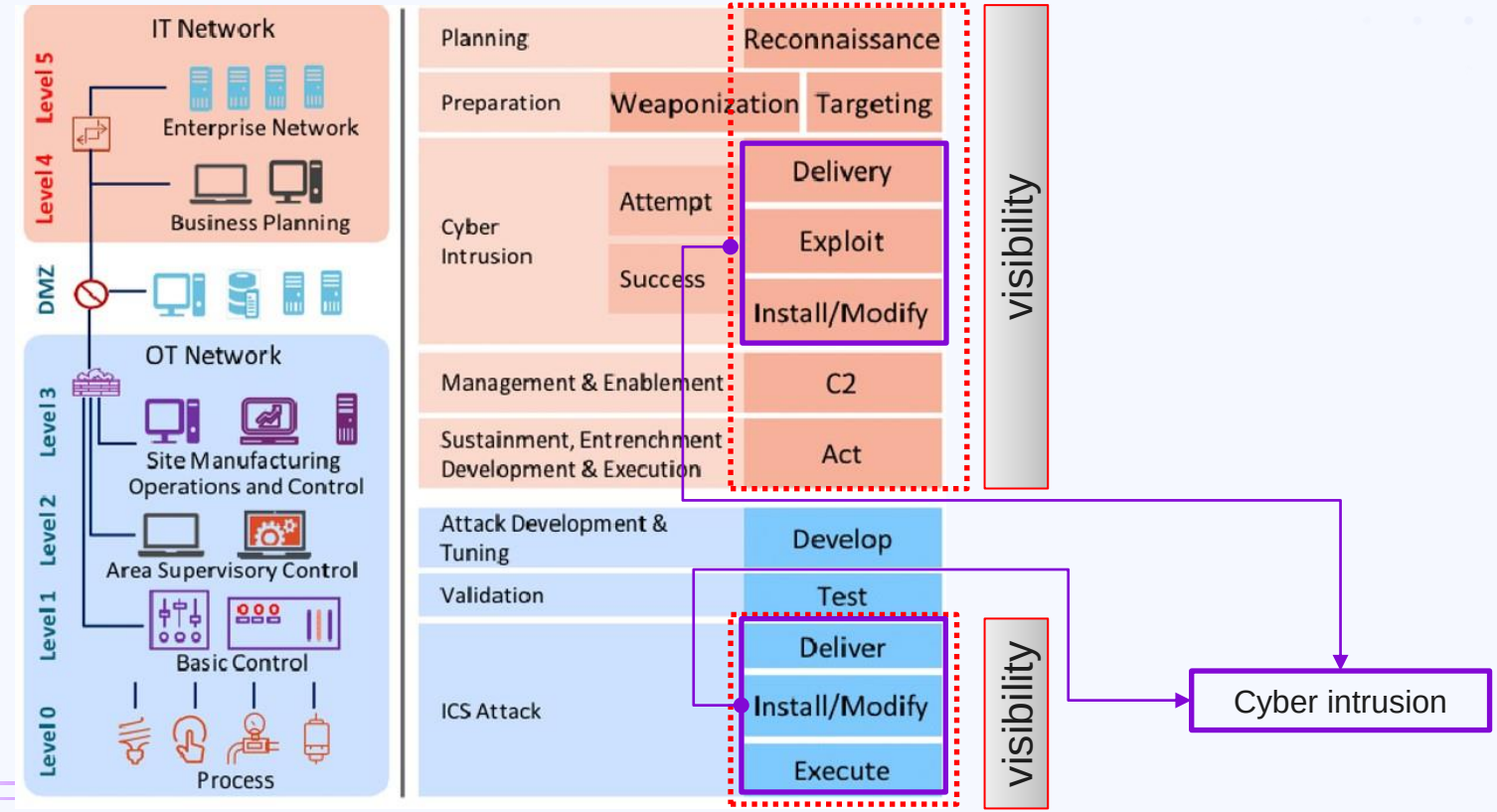


# ICS kill chain

**Stage 2**:
• **Attack Development and Tuning**: Creating ICS-tailored payloads using exfiltrated data.
• **Validation**: Testing in simulated ICS environments.
• **ICS Attack**: Execution of disruptive actions on OT systems (e.g., process manipulation).



# ICS kill chain in OT Architecture



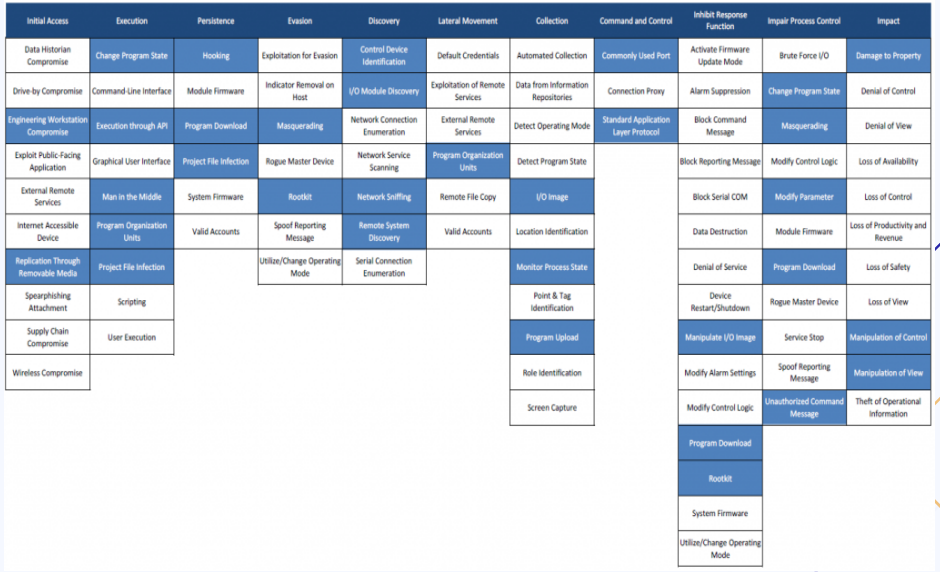
# ATT&CK for ICS

knowledge base and TTP catalogue of attacks organizing adversary behaviors into:

- **Tactics (TA IDs)**: 12 high-level "why" objectives, representing stages in the ICS Attack Lifecycle (from initial foothold to disruption).
- **Techniques (T IDs)**: Specific “what" methods to achieve tactics, with 85 total as of the latest matrix.
    - **Sub-Techniques**: Granular variations within techniques (e.g., T0859.001 for specific protocol exploits).
- **Procedures**: Real-world examples tied to threat groups, campaigns, or software (e.g., Stuxnet's use of T0885 for PLC firmware modification)(how).
- **Mitigations (M IDs)**: Countermeasures, such as M0951 (Update Software) to patch vulnerabilities in ICS devices, often requiring operational downtime planning.
- **Groups, Campaigns, and Software**: Over 20 adversary groups (e.g., Sandworm) and software (e.g., Industroyer) mapped to TTPs.

# Why This Matters for ICS Defenses

- Tactics → guide overall strategy (e.g., segment IT/OT to block lateral movement)
- Techniques → inform detection rules (e.g., monitor for anomalous PLC writes via tools like Snort for ICS protocols)
  Procedures→ enable threat hunting (e.g., check for Stuxnet-like USB artifacts in air-gapped environments).



| Initial Access | Execution | Persistence | Evasion | Discovery | Lateral Movement | Collection | Command and Control | Inhibit Response Function | Impair Process Control | Impact |
|---|---|---|---|---|---|---|---|---|---|---|
| Data Historian Compromise | Change Program State | Hooking | Exploitation for Evasion | Control Device Identification | Default Credentials | Automated Collection | Commonly Used Port | Activate Firmware Update Mode | Brute Force I/O | Damage to Property |
| Drive-by Compromise | Command-Line Interface | Module Firmware | Indicator Removal on Host | I/O Module Discovery | Exploitation of Remote Services | Data from Information Repositories | Connection Proxy | Alarm Suppression | Change Program State | Denial of Control |
| Engineering Workstation Compromise | Execution through API | Program Download | Masquerading | Network Connection Enumeration | External Remote Services | Detect Operating Mode | Standard Application Layer Protocol | Block Command Message | Masquerading | Denial of View |
| Exploit Public-Facing Application | Graphical User Interface | Project File Infection | Rogue Master Device | Network Service Scanning | Program Organization Units | Detect Program State | | Block Reporting Message | Modify Control Logic | Loss of Availability |
| External Remote Services | Man in the Middle | System Firmware | Rootkit | Network Sniffing | Remote File Copy | I/O Image | | Block Serial COM | Modify Parameter | Loss of Control |
| Internet Accessible Device | Program Organization Units | Valid Accounts | Spoof Reporting Message | Remote System Discovery | Valid Accounts | Location Identification | | Data Destruction | Module Firmware | Loss of Productivity and Revenue |
| Replication Through Removable Media | Project File Infection | | Utilize/Change Operating Mode | Serial Connection Enumeration | | Monitor Process State | | Denial of Service | Program Download | Loss of Safety |
| Spearphishing Attachment | Scripting | | | | | Point & Tag Identification | | Device Restart/Shutdown | Rogue Master Device | Loss of View |
| Supply Chain Compromise | User Execution | | | | | Program Upload | | Manipulate I/O Image | Service Stop | Manipulation of Control |
| Wireless Compromise | | | | | | Role Identification | | Modify Alarm Settings | Spoof Reporting Message | Manipulation of View |
| | | | | | | Screen Capture | | Modify Control Logic | Unauthorized Command Message | Theft of Operational Information |
| | | | | | | | | Program Download | | |
| | | | | | | | | Rootkit | | |
| | | | | | | | | System Firmware | | |
| | | | | | | | | Utilize/Change Operating Mode | | |

# Demo: MITRE ATTCK for Hunters

## Exercise: Using Matrix & navigator

- **APT33** is a suspected Iranian threat group that has carried out operations since at least 2013. The group has targeted organizations across multiple industries in the United States, Saudi Arabia, and South Korea, with a part

- **Sandworm Team:** Sandworm Team is a destructive threat group that has been attributed to Russia's General Staff Main Intelligence Directorate (GRU) Main Center for Special Technologies (GTsST) military unit 7445

02

# OT/ICS Network Visibility & Intrusion Detection Basics

Enabling OT Networks to detect cyber intrusions

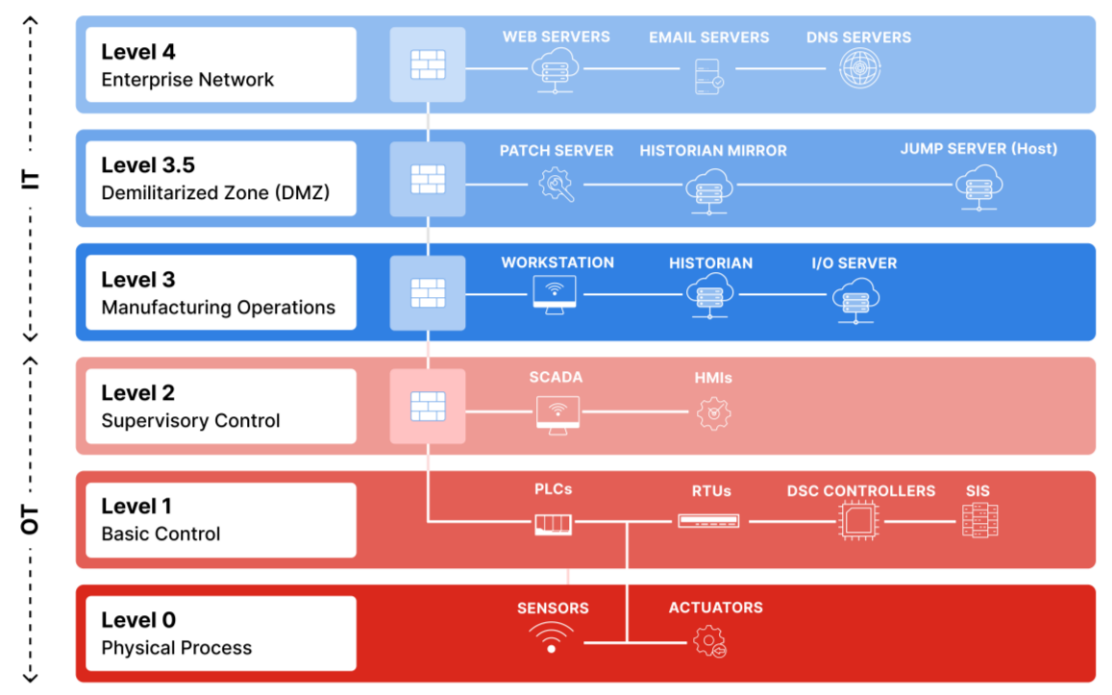
# Understanding Purdue Model

- Key Components: Programmable Logic Controllers (PLCs), Human-Machine Interfaces (HMIs), Remote Terminal Units (RTUs), SCADA server, Historian, and Engineering Workstation.

- Network Segmentation: The OT zone is separated from the IT zone by a firewall to restrict and monitor traffic flow.

- Remote Access: Access is via a secure gateway enforcing authentication and encrypted communications to isolate OT from internet threats.

# Purdue Model of OT Security




The Purdue Model provides clear boundaries between OT zones, ensuring segmentation and security. logical layers (0–4) separate IT and OT domains, limiting lateral movement and enforcing defense-in-depth.

Threat detection can be enabled by IDS/IPS, anomaly detection, and network monitoring at each level.

# Purdue Model of OT Security

## Level 0-1: Field Devices & Control System

Sensors, actuators, PLCs, and other physical devices.

Vulnerable to attacks, often exposed to direct access.

Threat Detection: Use of Industrial IDS/IPS (Intrusion Detection/Prevention Systems), physical device monitoring.

# Purdue Model of OT Security

**Level 2: Control Network:**

•**SCADA** systems and **HMIs**.

•Manages critical industrial control functions.

• **Threat Detection:** Implement **network traffic analysis** and **endpoint monitoring** (e.g., SIEM tools).

# Purdue Model of OT Security

## Level 3: Operations

•Operations management, MES (Manufacturing Execution Systems).

•Contains data needed for production planning and execution.

•**Threat Detection:** Secure access and monitor **device communication** with **firewalls** and **data encryption**.

# Purdue Model of OT Security

## Level 4: Business Network

•Corporate IT systems, ERPs, and cloud services.
•Interfaces with OT to enable analytics and reporting.
•**Threat Detection: Vulnerability scans**, **antivirus**, and **file integrity monitoring**

# Purdue Model of OT Security

## Level 4: Business Network

•Corporate IT systems, ERPs, and cloud services.
•Interfaces with OT to enable analytics and reporting.
•**Threat Detection: Vulnerability scans**, **antivirus**, and **file integrity monitoring**

# Purdue Model of OT Security

## Level 5: Enterprise IT Systems

•Centralized data management and business applications.

•Interfaces and exchanges data with lower OT levels.

•**Threat Detection: Advanced firewall setups**, **traffic encryption**, **data loss prevention** systems.

# ICS Network Visibility Components

## Achieving real-time visibility in OT networks is critical for timely detection and response to threats

- **Asset Discovery** Catalog all OT devices (PLCs, HMIs, RTUs, servers).
- Use passive monitoring (SPAN/TAP) to avoid disrupting sensitive systems.
- **Network Segmentation and Traffic Control** industrial firewalls and zone-based architecture to limit traffic paths and reduce attack surface.
- **Protocol Awareness** Deploy OT-aware Intrusion Detection Systems (IDS), or sensors that understand OT protocols (Modbus, DNP3, OPC UA) and monitor for anomalous behaviors.
- **Centralized Log Collection and Analysis** Forward logs from OT devices, firewalls, and IDS to a Security Information and Event Management (SIEM) system./Employ correlation rules tuned for OT threat indicators.
- **Threat Hunting Capability** Develop playbooks for manual threat hunting using threat intelligence and behavioral analytics.
- Note: Use active scanning tools cautiously to avoid disrupting OT process controls.

# Methods of detection

Attacks in ICS often exploit **trust and predictability**. Detection should cover:

**Signature-based Detection (Known Threats):** Detects malware (e.g., Stuxnet, Industroyer) and exploits with known patterns. Tools: Snort/Suricata with ICS-specific rules.

**Anomaly-based Detection (Unknown Threats):** Detect unusual commands, traffic spikes, or new device communications. Example: An unexpected “Write” command sent to a PLC at midnight.

**Stateful Protocol Analysis/Behavioral Detection:** compares observed behavior to normal protocol states. detects attacks that misuse legitimate protocols (like DNS tunneling). In ICS Environments Behavioral Detection Model normal process values (temperatures, pressures).Alert if commands push process out of safe bounds.

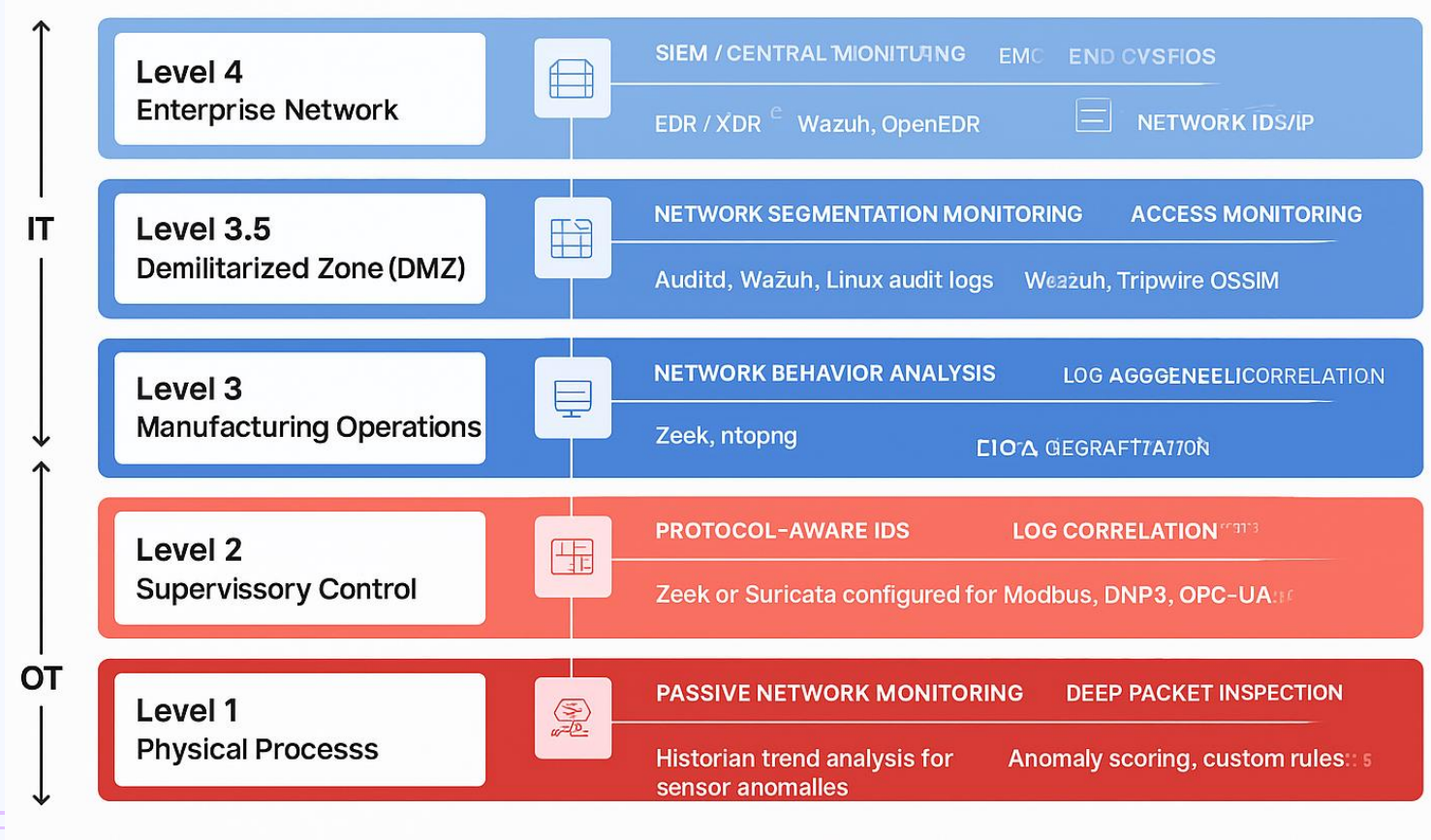
# ICS Attack Techniques(ATT&CK View)

| Attack | ATT&CK ID |
|---|---|
| **Command Injection:** Malicious writes to PLC memory/registers. | T0855 (Unauthorized Command Message), T0831 (Manipulation of Control) |
| **Replay Attacks:** Sending old valid data to trick operators. | T0831 (Manipulation of Control), T0830 (Adversary-in-the-Middle) |
| **Man-in-the-Middle:** Altering sensor readings or operator commands | T0830 (Adversary-in-the-Middle) |
| **Rogue Device Insertion:** A new laptop or controller connected to the network. | T0848 (Rogue Master), T1200 (Hardware Additions) |
| **Protocol Misuse:** Abusing functions like “Stop PLC” or “Firmware Update”. | T0800 (Activate Firmware Update Mode), T0857 (System Firmware), T0855 (Unauthorized Command Message), T0875 (Change Program State) |

# Threat Detection Arsenals

| Purdue | Zone | Key Tools | Primary Goal |
|---|---|---|---|
| L4 | Enterprise IT | ELK, Wazuh, Suricata, MISP | Stop IT-origin threats before OT pivot |
| L3.5 | DMZ | Suricata, Zeek, File Integrity, Jump Host Monitoring | Secure IT–OT boundary |
| L3 | Operations | Zeek, Graylog/ELK, TheHive, OT IDS | Detect internal movement, abnormal data |
| L2 | Supervisory | SCADA logs, Zeek/Suricata (ICS rules) | Detect HMI or SCADA misuse |
| L1 | Control | PLC traffic monitoring, firmware checks(Sysmon + Wazuh, Auditd (Linux)) | Stop unauthorized commands |
| L0 | Physical | Sensor anomaly,SIS monitoring, custom rules, pcaps, wireshark | Detect physical process manipulation |

# Purdue under detection umbrella



# 03

# Detection and hunting Using Open-source-tool

Getting your hand dirty

# Tools Detection Approaches

Signature-based → (Snort/Suricata)

Anomaly-based → (Zeek)

Behavior-based → ELK/Graylog/

- Baselining & analyzing and Hunt suspicious behavior (ML/EDR/SIEM)

# Detection Tools

Suricata/Snort: Signature-based IDS for ICS threats
Zeek: Protocol analysis & anomaly detection
ELK/Graylog: Centralized logging and dashboards
Wazuh Edge/endpoint monitoring
MISP: Threat intelligence integration

# Hands-On Exercise 1:

## Modbus Traffic Analysis Example

Examining **Modbus** traffic allows differentiation between normal and anomalous activity critical for ICS security.

- Provide PCAP files with ICS traffic
- Task: Identify abnormal PLC commands and unauthorized HMI login attempts

# Hands-On Exercise 2:

## Create ics rule in Suricata and sending logs to ELK

Pipeline: Collect → Detect → Alert → Investigate

Configuring Suricata rules for Modbus anomalies

create a simple detection rule in Suricata for “STOP PLC command”

Validate alerts in ELK dashboard

Sending Zeek logs into ELK and building a detection dashboard

# Exercise: Suricata for detection and hunt

Open /etc/suricata/suricata.yaml and check/enable app-layer protocol detection. Key points:Ensure app-layer is enabled (default).Set protocols: or per-proto config to allow Modbus, DNP3, ENIP/CIP, etc., as needed.

```
app-layer:
  protocols:
    modbus:
      enabled: yes
      detection-ports:
        dp: [ 502 ]
    dnp3:
      enabled: yes
      detection-ports:
        dp: [ 20000 ]
    enip:
      enabled: yes
    iec104:
      enabled: yes
```

/etc/suricata/suricata.yaml

```
outputs:
  - eve-log:
      enabled: yes
      filetype: regular
      filename: /var/log/suricata/eve.json
      types:
        - alert
        - flow
        - anomaly
        - modbus
        - dnp3
        - enip
        - iec104
```

Alert into eve.json

# Exercise: Suricata for detection and hunt

Drop in /etc/suricata/rules/local.rules

```
alert tcp any any -> any 5020 (msg:"Modbus Write Single Coil"; app-layer-protocol:modbus; modbus.func_code:5; sid:1000002; rev:1;)

alert tcp any any -> any 5020 (msg:"Modbus Read Holding Registers"; app-layer-protocol:modbus; modbus.func_code:3; sid:1000003; rev:1;)
```

```
sudo suricata-update
sudo systemctl restart suricata
```

Dataset: CIC Modbus Dataset 2023
Dataset: 4SICS ICS Lab PCAP Files

# Exercise: Suricata for detection and hunt

Open /etc/suricata/suricata.yaml

```
app-layer:
  protocols:
    http: yes
    modbus: yes
    dnp3: yes
    enip: yes
    # iec104 may not have full keyword support — enable generic tcp/app detection
```

# Exercise: Zeek for detection and hunt

Zeek as NTA/Behavior-based detection to analysis traffic and ICS-specific attacks. Generate .log files for analyzing in SIEM

Install zkg (Zeek Package Manager) and packages

Zeek Script/parsers for ICS:

```
zkg refresh
zkg install icsnpp-modbus
```

```
zkg refresh
zkg install icsnpp-s7comm
```

```
/opt/zeek/share/zeek/site/packages/
├── icsnpp-modbus/
│   ├── scripts/
│   ├── README.md
│   └── …
├── icsnpp-dnp3/
│   ├── scripts/
│   └── …
├── icsnpp-enip/
├── icsnpp-s7comm/
└── zeek-plugin-bacnet/
```

*.zeek scripts defining events, records, and logging.*

# Exercise: Zeek for detection and hunt

customize Zeek scripts

Extend from *local.zeek file*
Add your own .zeek script file

Example:
detect Modbus write function codes

from */opt/zeek/share/zeek/site/local.zeek* *create an template*
Add your own script file
*/opt/zeek/share/zeek/site/custom/detect_modbus.zeek*

and in include in local.zeek
*@load custom/detect_modbus*

*check for loading in zkg*
*cat /opt/zeek/share/zeek/site/packages/icsnpp-modbus/__load__.zeek*

```
@load packages/icsnpp-modbus

event zeek_init()
    {
    print "Custom ICS detection loaded!";
    }

event Modbus::request(c: connection, req: Modbus::Request)
    {
    if ( req$function_code == 16 ) {
        NOTICE([$note=Notice::Policy,
                $msg=fmt("Modbus write multiple registers from %s", c$id$orig_h),
                $conn=c]);
    }
    }
```

# Exercise: Zeek for detection and hunt

customize Zeek scripts

Extend from *local.zeek file*
Add your own .zeek script file
Test it

- On pcaps:

```
sudo /opt/zeek/bin/zeek -C -r
/path/to/test_modbus.pcap
local.zeek
```

- Live capture(real-interface):

```
sudo /opt/zeek/bin/zeek -i eth0 local.zeek
# or if using zeekctl:
sudo /opt/zeek/bin/zeekctl deploy
```

# Exercise: Zeek for detection and hunt

Zeek as NTA/Behavior-based detection to analysis traffic and ICS-specific attacks. Generate .log files for analyzing in SIEM

Zeek-data sets: test traces
Zeek Parsers: ICSnpp

Zeek Script/parsers for ICS:

```
zkg refresh
zkg install icsnpp-modbus
```

```
zkg refresh
zkg install icsnpp-s7comm
```

Logs stored in → /opt/zeek/logs/current/

# Exercise: Wazuh for detection and hunt

Wazuh can detect and monitor OT/ICS traffic (like Modbus, DNP3, BACnet, etc.) using:

- Using Wazuh custom decoders and rules  (act as EDR/FIM on endpoint)
- Collecting logs from network monitoring tools (e.g. Zeek, Suricata, Snort) & alerts for protocol anomalies or specific events

## Main Components

| Component | Purpose | Key Path |
|---|---|---|
| **Decoders** | Parse protocol fields from logs | /var/ossec/etc/decoders.d/ |
| **Custum Rules** | Trigger alerts based on decoded data | /var/ossec/etc/rules.d/ |
| **main config** | Enable modules, inputs, or log sources | /var/ossec/etc/ossec.conf |
| **Agent config** | Define local log collection | /var/ossec/etc/ossec.conf on agent |
| **Logs for testing** | Store log samples to simulate events | /var/ossec/logs/<br>/var/ossec/logs/alerts/alerts.log |
| **Local rules** | | /var/ossec/etc/rules/local_rules.xml |

# Exercise: Wazuh for detection and hunt

Example: Modbus Detection Setup

Step 1: Add a decoder File: /var/ossec/etc/decoders.d/modbus_decoders.xml

```xml
<decoder name="modbus">
  <program_name>suricata</program_name>
  <prematch>modbus</prematch>
  <regex>src_ip=(\S+) dst_ip=(\S+) func_code=(\d+)</regex>
  <order>src_ip, dst_ip, func_code</order>
</decoder>
```

Step 2: Add a rule File: /var/ossec/etc/rules.d/modbus_rules.xml

```xml
<group name="modbus,network,">
  <rule id="100500" level="7">
    <decoded_as>modbus</decoded_as>
    <description>Modbus command detected</description>
  </rule>

  <rule id="100501" level="10">
    <if_sid>100500</if_sid>
    <match>func_code=5</match>
    <description>Modbus Write Single Coil Command Detected</description>
  </rule>
</group>
```

# Exercise: Wazuh for detection and hunt

Example: Modbus Detection Setup

## Step 3:Test the Rules

Restart Wazuh manager

sudo systemctl restart wazuh-manager

## Create a test log entry :

echo 'modbus src_ip=192.168.1.10 dst_ip=192.168.1.20 func_code=5' >> /var/ossec/logs/test_modbus.log

Check alerts

sudo tail -f /var/ossec/logs/alerts/alerts.log

sudo cat /var/ossec/logs/alerts/alerts.json | grep modbus

# Exercise: Wazuh for detection and hunt

**Example: OT/ICS Detection Workflow**

1. **Deploy a network sensor** (e.g. Suricata or Zeek) to capture ICS protocols.
2. **Forward logs** (e.g. JSON via syslog or filebeat) to Wazuh.
3. **Create custom decoders** to parse ICS protocol fields.
4. **Write rules** to detect anomalies or specific commands.
5. **Reload Wazuh** and **test alerts**.

# Exercise: KIBANA Query & dashboard OTCICS

build OT/ICS protocol detection with the Elastic stack (ELK: Elasticsearch + Logstash/Beats + Kibana)

Export logs in structured JSON (Zeek logs, Suricata eve.json)

- Suricata produces eve.json containing alerts
- Zeek generates modbus.log, dnp3.log, iec104.log (depending on plugins)

Ship logs with Filebeat (or Logstash) into Elasticsearch

create Kibana visualizations & detection rules
(Elastic Security / SIEM).

# Exercise: KIBANA Query & dashboard OTCICS

Ship logs with Filebeat (or Logstash) into Elasticsearch

- Enable Filebeat Suricata module (parses eve.json into ECS)

```
# enable suricata module
sudo filebeat modules enable suricata

# configure module (edit /etc/filebeat/modules.d/suricata.yml)
# point to Suricata eve.json path, e.g. /var/log/suricata/eve.json

# setup index templates and dashboards (one-time)
sudo filebeat setup --dashboards

# start filebeat
sudo systemctl enable --now filebeat
```

# Exercise: KIBANA Query & dashboard OTCICS

Ship logs with Filebeat (or Logstash) into Elasticsearch

- Ingest Zeek logs with Filebeat or Logstash
  - enable zeek module in Filebeat or
  - use filebeat.inputs for custom path and ingest_pipeline.

# Exercise: KIBANA Query & dashboard OT/ICS

- Use Filebeat / Suricata dashboards as baseline (filebeat setup --dashboards).
- Elastic Create visualizations per protocol:
  - connections by suricata.app_proto or Zeek service/protocol fields.
- Use Elastic Security detection rules (KQL or threshold-based) to generate alerts.

Example KQL queries

```
suricata.eve.alert.signature : "MODBUS" or suricata.eve.alert.signature : "*modbus*"
```

Suricata Modbus alerts

```
event.dataset : "zeek.modbus" and modbus.func_code : *
```

Zeek Modbus activity

# Exercise: KIBANA Query & dashboard OT/ICS

- Use Filebeat / Suricata dashboards as baseline (filebeat setup --dashboards).
- Elastic Create visualizations per protocol:
  - connections by suricata.app_proto or Zeek service/protocol fields.
- Use Elastic Security detection rules (KQL or threshold-based) to generate alerts.

Example detection rule:

Rule type: Custom query rule
Index: suricata-*, zeek-*
KQL:
(event.category: "network" AND (suricata.eve.alert.signature : "*modbus*" OR event.dataset : "zeek.modbus"))

Conditions: trigger when count >= 5 in 5 minutes for a given source.ip (to detect scanning/brute actions).

# Exercise: KIBANA Query & dashboard OT/ICS

Useful example detections:

- Unusually large number of Modbus Function 0x05/0x06 (write coil/register) from single source → possible unauthorized writes.
  - Query: count by source.ip where modbus.func_code in [5,6,15,16] over 5m
- DNP3 link-layer resets / forbidden commands → match DNP3 frame types.
- New/unknown devices talking ICS protocols → baseline expected IPs and alert on unknown device_id or unit_id.
- Protocol mismatch on expected ports (e.g., Modbus on non-502 port) → suspicious.

# Threat Hunting in OT

The goal is not to *find alerts* but to **uncover evil**.

"We are compromised, we just don't know it yet."

use hypotheses to guide their search through data, looking for Tactics, Techniques, and Procedures (TTPs) associated with known threat actors(APTS) targeting industrial systems.

Key principles:

- Proactive, Not Reactive: You are seeking out threats, not waiting for an alarm.
- Hypothesis-Driven: Every hunt begins with a testable statement. (e.g., "An adversary is using the engineering workstation as a pivot point to manipulate PLC logic.")
- Human-Centric: It relies on skilled analysts' intuition, knowledge of the environment, and understanding of adversary behavior.
- Data-Centric: It requires deep visibility into network traffic, device logs, and process data

# Hunting Loop in OT (1)

1.**Hypothesis Formation:** This is the starting point. Hypotheses are based on:

1. **Threat Intelligence:** MITRE ATT&CK for ICS, reports on groups like Xenotime, Chimera, etc.
2. **Asset & Network Criticality:** "What is the most dangerous thing an attacker could do to our gas compressor / reactor / grid feeder?"
3. **Known Vulnerabilities:** New CVEs affecting your specific PLC or HMI models.
4. **Behavioral Anomalies:** Unexplained process fluctuations or operator complaints.

# Hunting Loop in OT (2)

**2. Data Collection & Investigation:** The hunter gathers and queries data to test the hypothesis. Key OT data sources include:

1. **Network Traffic:** Flow data (NetFlow) and full packet capture (PCAP) from SPAN ports on OT network switches.
2. **Host Logs:** Windows event logs from HMIs and Engineering Workstations.
3. **Process Logs:** Historian data (OSIsoft PI, Aveva), alarm logs from the DCS/SCADA.
4. **Asset Inventory:** A detailed list of all OT devices with their make, model, firmware, and network address.

# Hunting Loop in OT (3)

**3. Trigger & Triage:** The investigation reveals a "trigger"—a piece of evidence that warrants deeper analysis. This could be:

1. An unknown IP address communicating with a critical PLC.
2. A command sequence sent at an unusual time.
3. A change in a PLC's ladder logic that wasn't part of a sanctioned change control.

# Hunting Loop in OT (4)

**5. Resolution:** The hunter determines if the trigger is a true positive (a real threat) or a false positive (benign activity).

- **True Positive:** Initiate the Incident Response (IR) process immediately. **Coordination with operations is essential before any containment action.**
- **False Positive:** Document the finding. This improves future hunts and can be used to tune automated alerts.

# Hunting Loop in OT (5)

**5. Knowledge Enrichment:** Regardless of the outcome, the hunt generates valuable knowledge.

1. **New Detection Rules:** Create a SIEM or IDS signature for the discovered TTP.
2. **Improved Hygiene:** Identify a misconfiguration or weak security practice that needs remediation.
3. **Refined Hypotheses:** Inform the next round of hunting.

# Casestudy: compromised EWS

- Hypothesis: "An adversary has compromised the Engineering Workstation and is periodically, subtly altering the PLC logic
  - Why: The Engineering Workstation is the "keys to the kingdom" – it can program all PLCs.
- This TTP is documented in MITRE ATT&CK for ICS (T0833 - Modify Controller Logic).

# Exercise: compromised EWS

- What Data We Need?
- We can't just scan the PLC; we need to correlate data.
- Network Data: Full packet capture on the control network segment.
- Host Data: Windows Event Logs from the Engineering Workstation (especially logons and process execution).
- Process Data: The Process Historian (OSIsoft PI) logging the Mixing Vat's temperature, pressure, and mix-time.

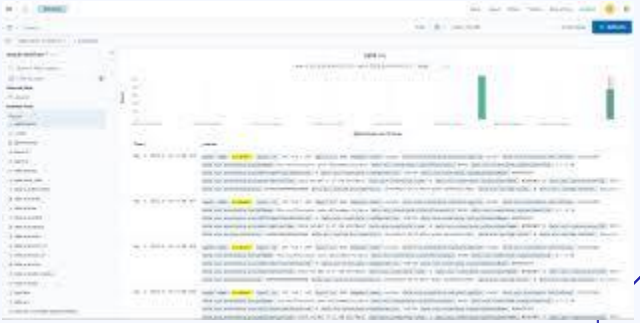
Hunting tools: Zeek logs & ELK queries

# Exercise: compromised EWS

- The Investigation - Finding the Trigger
- "We started by looking for the simplest evil: Was the engineering software used at a strange time?"
- Query: Search Windows Event Logs for TIA Portal (Siemens PLC programming software) execution.
- Finding: We found it was launched at 2:17 AM on a Saturday. No maintenance was scheduled. This is our trigger.

# Exercise: compromised EWS

- Corroborating the Evidence - Network Footprint
- "Next, we went to the network data. Did the workstation talk to the PLC at that time?"
- Query: Filter packet capture for traffic between the Engineering Workstation and the Mixing Vat PLC IP address around 2:17 AM.
- Finding: A clear, sustained communication session using the S7Comm (Siemens) protocol for over 5 minutes. This is not a simple polling request; it's a download session.



# Exercise: The Smoking Gun - Process Impact

"So we know someone changed the logic. But what did they change? We had to look at the physical process."

Query: Extract from the Process Historian the Mix_Time parameter for the vat for the week before and after the incident.

Finding: The Mix_Time setpoint was silently changed from 120.0 seconds to 118.5 seconds during the 2:17 AM session. This 1.5-second reduction was just enough to create a minor imperfection, explaining the quality control failures.

# Exercise: Resolution and Impact

- **True Positive Confirmed:** Malicious, unauthorized logic modification.
- **Immediate Action:**
  - **COORDINATED WITH OPERATIONS FIRST!** We did not disconnect anything blindly.
  - Operations took the vat offline safely during a pre-planned transition.
  - The PLC logic was restored from a known-good backup.
  - The Engineering Workstation was isolated and forensically imaged.
- **Root Cause:** Compromised credentials for a service account used on the Engineering Workstation.

# Exercise: Resolution and Impact

- **True Positive Confirmed:** Malicious, unauthorized logic modification.
- **Immediate Action:**
  - **COORDINATED WITH OPERATIONS FIRST!** We did not disconnect anything blindly.
  - Operations took the vat offline safely during a pre-planned transition.
  - The PLC logic was restored from a known-good backup.
  - The Engineering Workstation was isolated and forensically imaged.
- **Root Cause:** Compromised credentials for a service account used on the Engineering Workstation.

# Exercise: Hunting at L1 – control layer

| Component | Purpose |
| --- | --- |
| **Zeek** | Network protocol analysis (Modbus, DNP3) |
| **Suricata** | Intrusion detection (ICS ruleset) |
| **ELK Stack (Elastic, Logstash, Kibana)** | Visualization and correlation |
| **Wazuh** | SIEM agent + FIM alerts |
| **PCAP Dataset** | “ICS_attack_lab1.pcap” (Modbus unauthorized write example) |
| **Workstation** | Analyst VM with Zeek + ELK |
| **Network Range** | 10.10.10.0/24 (Simulated Control Network) |

# Bonus: Playbooks for OT

Incident response playbooks for OT security incidents are structured guides that provide specific steps for detecting, containing, eradicating, and recovering from cyber incidents in operational technology environments.

# OT Incident Response Playbook template

1.Initiating Conditions
1. Define the specific triggers that start the playbook, such as detection of malware in OT network, anomalous PLC commands, unauthorized remote access, or safety system alerts.

2.Roles and Responsibilities
1. Assign clear roles, including Incident Manager (overall coordination), Technical Lead (investigation and remediation), Communications Lead (internal/external notifications), and OT Operations Liaison (ensure process safety).

3.Initial Containment
1. Isolate affected OT segments to prevent lateral movement.
2. Preserve evidence by capturing network logs and device states.
3. Restrict remote access and disable suspect accounts or sessions.

4.Investigation and Analysis
1. Identify the attack vector and scope through forensic analysis of logs, network traffic, and device behavior.
2. Verify integrity and operational status of critical devices such as PLCs, SIS, and HMIs.
3. Engage OT process experts to assess impact on safety and production.

5.Eradication and Remediation
1. Remove malware, unauthorized access points, and backdoors.
2. Apply patches or configuration changes to close exploited vulnerabilities.
3. Restore affected devices from secure backups where necessary.

6.Recovery
1. Gradually reintroduce isolated systems to the network, closely monitoring for anomalies.
2. Validate safe and normal operation of control systems and safety functions.
3. Communicate system status and timelines to stakeholders.

7.Post-Incident Activities
1. Conduct thorough postmortem to determine root cause.
2. Update defenses and detection capabilities based on lessons learned.
3. Review and revise the playbook and training as needed to improve future responses.

# Conclusions

# Thanks!

## Do you have any questions?

**Email:** Mehdi.Sayad@yahoo.com

**Whatsup:** +98 935 936 9669

**Linkein:** https://www.linkedin.com/in/msayyad

**GitHub:** https://github.com/Hex0r1/ISCISC2025-workshop