Grid-SIEM

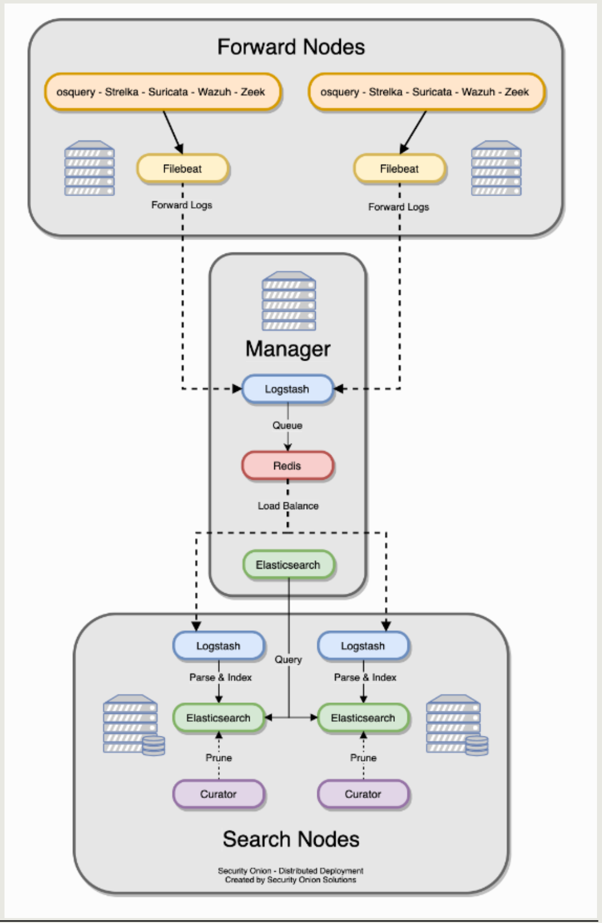
SDMAY24-29

Agenda

- SecurityOnion Overview
- Moving SecurityOnion data to PyTorch
- Machine Learning Options
- PowerCyber Grid Structure
- Team Questions

Security Onion

- We plan to implement a distributed architecture as talked about and the architecture matches up with the diagrams of the CPS Security DER Modbus Testbed.
- The nodes that we plan to implement are a Manager node, Search node and Forward nodes, the forward nodes are the sensors and will be required to collect logs and data from the DER Clients. The Manager node is the central location for the SIEM and will be the center point that users will view the SOC (Security Onion Console) from. The Search node takes data from the Redis queue in the Manager node to parse it and search through it when a user uses Elasticsearch on the Manager node to search for results.
- Hardware Requirements
  - Manager – 16 GB RAM, 4-8 CPU Cores, 1TB Storage
  - Forward – 12 GB RAM, 4 CPU Cores, 200 GB Storage
  - Search – 16-64 GB RAM, 4-8 CPU Cores, 200GB Storage
- Logstash is the current beta machine learning tool for detecting logs relating to login attempts and failures. Our project can pull logs from the Manager node in a similar way for our machine learning component.



Security Onion Data to PyTorch

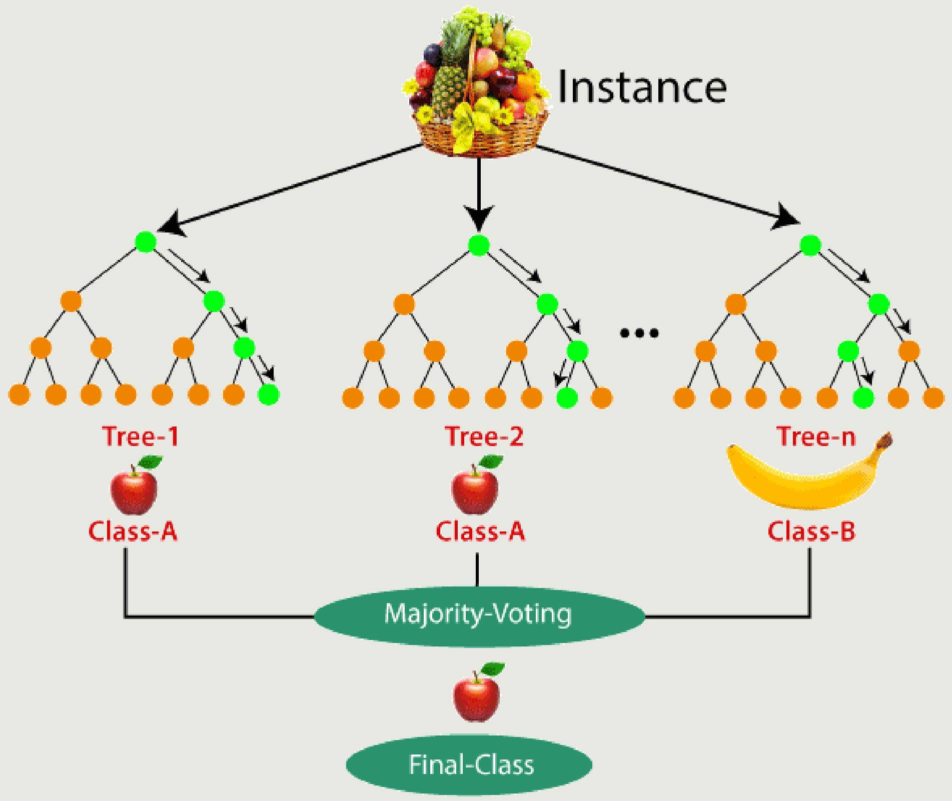
- Jupyter notebook
- Security Onion logs in JSON
    - ElasticSearch query to extract relevant data
- Clean the data using pandas
    - Pandas specific JSON read functionality
    - Missing values, normalize the data, etc.
- Integrate cleaned data to PyTorch workflow
    - Tensors
        - Multi-dimensional array data structure
        - Floating point, integer, Boolean
        - Allocate memory for each tensor
            - Amount of data to train?
- PyTorch machine learning model to evaluate

PyTorch Machine Learning

- Collect network traffic, logs, alerts, etc.
- Feature extraction
  - Determine necessary elements of the data
- Model Selection
  - Random Forest
    - Classification problem
    - Good for large data sets
    - Doesn't overfit the data
    - Aggregating the outputs of multiple random decision trees

Random Forest

- A collection of random decision trees
- Majority decision decides the outcome
- Randomness
  - Features
  - Uses only a sample of data each time a tree is fit
    - Uses random data (per tree)
    - There is never a single decision tree that has all rows and all features included
    - Means more variation between each tree and lessens chances of overfitting the data
- Hyperparameter tuning
  - Max depth - the number of questions before prediction is made
  - Number of estimators – number of decision trees in the forest
  - Max features – the number of columns in each decision tree
  - Bootstrapping
    - Randomly drawing samples from the data with replacement
      - Replacement – one data point can be used multiple times



Building the Random Forest

- Implement decision tree from scratch
  - Random sampling (bootstrapping) functionality
  - Recursively split data based on features
  - Choose best split criteria
  - Growing each tree to a specified depth
- Feature randomization
  - Need to randomly select splits for each branch of each decision tree
- Training the forest
  - Use a loop to train multiple decision trees where each has a different randomization
- Prediction
  - Majority vote
- Evaluation
  - Determine accuracy and precision

D-IDS for Cyber-Physical DER Modbus System

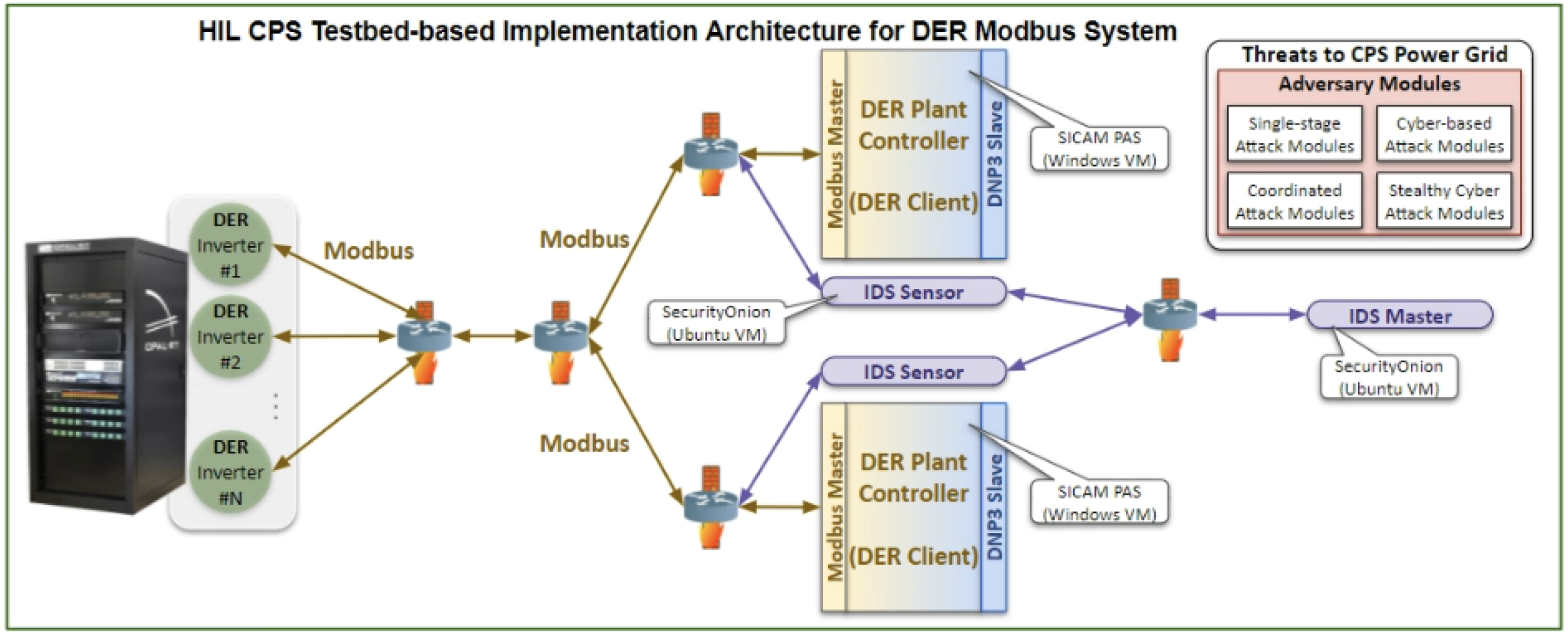


Fig. 6. Hardware in the Loop CPS Security DER Modbus Testbed

D-IDS for Cyber-Physical DER Modbus System

- As DERs/ICS increasingly become connected to each other and the internet, essentially becoming industrial IoT devices. There is a crucial need to secure these devices whose compromise by a malicious actor could have catastrophic consequences.
- Three main levels: DER-Management Systems (control level), DER Aggregators (middle level), DER Plant controllers (field level).
- Why does The California rule-21, Common Smart Inverter Profile [14], and DER communication standards recognize the Modus protocol as a widely accepted communication between DER field devices and DER plant controllers?
- It is essential to develop IDS methods tailored to DER Modbus communication devices.

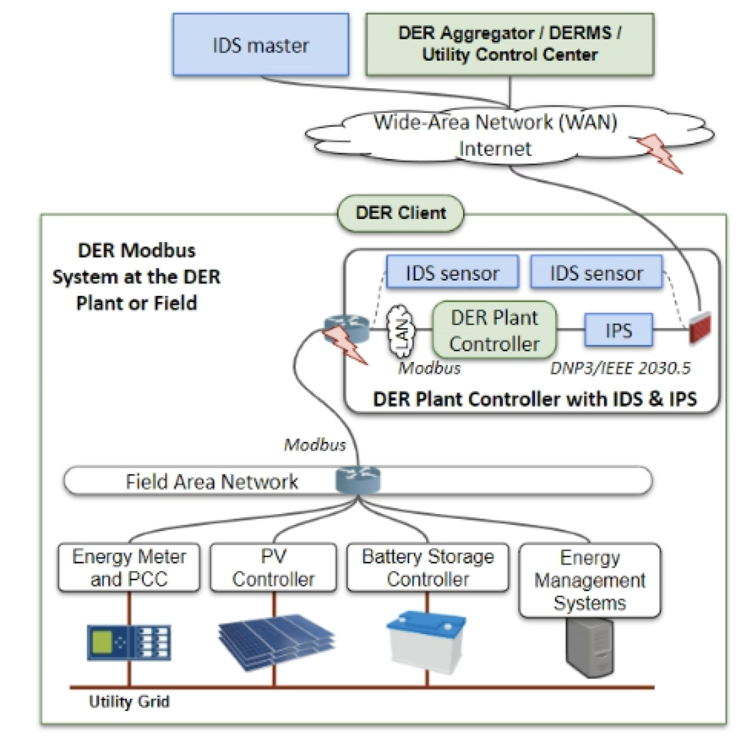

IDS master
DER Aggregator / DERMS / Utility Control Center
Wide-Area Network (WAN) Internet
DER Client
DER Modbus System at the DER Plant or Field
IDS sensor
IDS sensor
LAN
DER Plant Controller
IPS
Modbus
DNP3/IEEE 2030.5
DER Plant Controller with IDS & IPS
Modbus
Field Area Network
Energy Meter and PCC
PV Controller
Battery Storage Controller
Energy Management Systems
Utility Grid

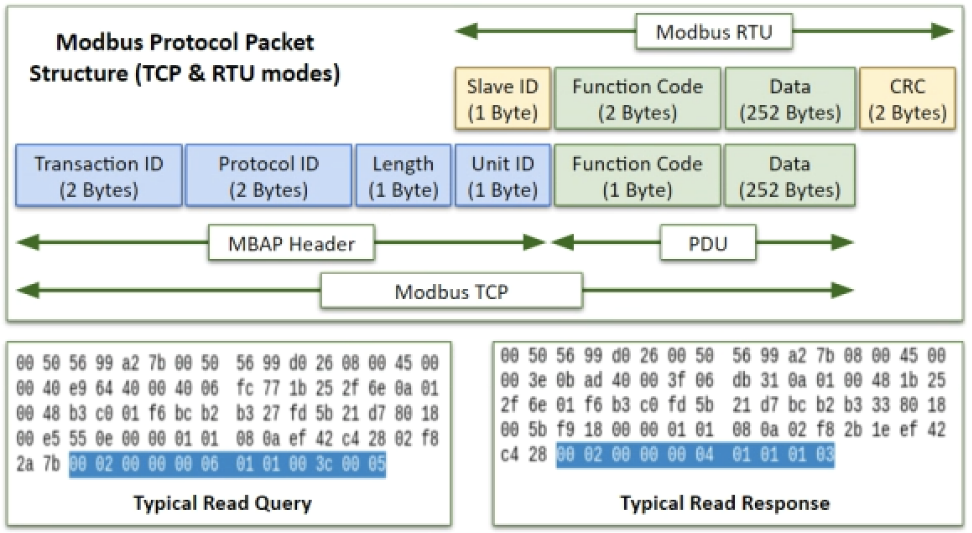
Fig. 2. Proposed D-IDS Architecture for DER Modbus Communication

D-IDS for Cyber-Physical DER Modbus System

- DER: Distributed Energy Resources
- CPSs: Cyber Physical Systems
- What: This paper proposes a distributed intrusion detection system (D-IDS) architecture and algorithms for detecting anomalies on the DER Modbus communication.
- How: The proposed IDS algorithm uses the model-based approach to develop Modus-specific IDS rule sets, which can enhance the detection accuracy of the anomalies either by data-integrity attacks or maloperation on cyber-physical DER Modbus devices.
- Why: The traditional information technology (IT) based cybersecurity technologies may have substantial false-positives and false-negatives with the continuous expansion of networked DER devices.
- Why was the native Modbus protocol designed to use a clear-text format?

D-IDS for Cyber-Physical DER Modbus System

- Modbus protocol developed by Schneider Electric.
- The Modus client responds to queries made by any legitimate or illegitimate Modus server who makes a request, creating a gap in the security of the DER device and infrastructure, and can be exploited by an adversary with malicious intent.
- As the Modbus protocol was built for faster and efficient peer-to-peer communication and did not include cybersecurity features, it is more vulnerable to cyber threats.



```
00 50 56 99 a2 7b 00 50  56 99 d0 26 08 00 45 00
00 40 e9 64 40 00 40 06  fc 77 1b 25 2f 6e 0a 01
00 48 b3 c0 01 f6 bc b2  b3 27 fd 5b 21 d7 80 18
00 e5 55 0e 00 00 01 01  08 0a ef 42 c4 28 02 f8
2a 7b 00 02 00 00 00 06  01 01 00 3c 00 05
```

```
00 50 56 99 d0 26 00 50  56 99 a2 7b 08 00 45 00
00 3e 0b ad 40 00 3f 06  db 31 0a 01 00 48 1b 25
2f 6e 01 f6 b3 c0 fd 5b  21 d7 bc b2 b3 33 80 18
00 5b f9 18 00 00 01 01  08 0a 02 f8 2b 1e ef 42
c4 28 00 02 00 00 00 04  01 01 01 03
```

Fig. 1. Modbus RTU and Modbus TCP Protocol Packet Structures