November 29th Update

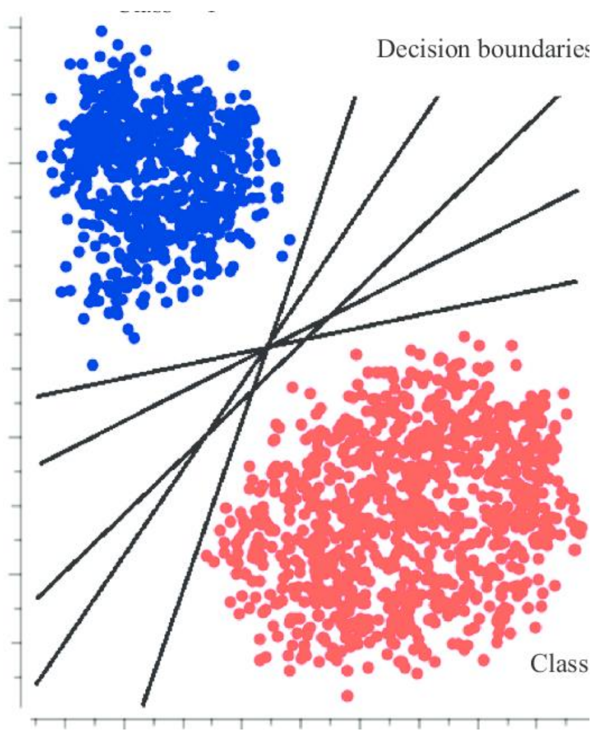
Grid-SIEM Group 29

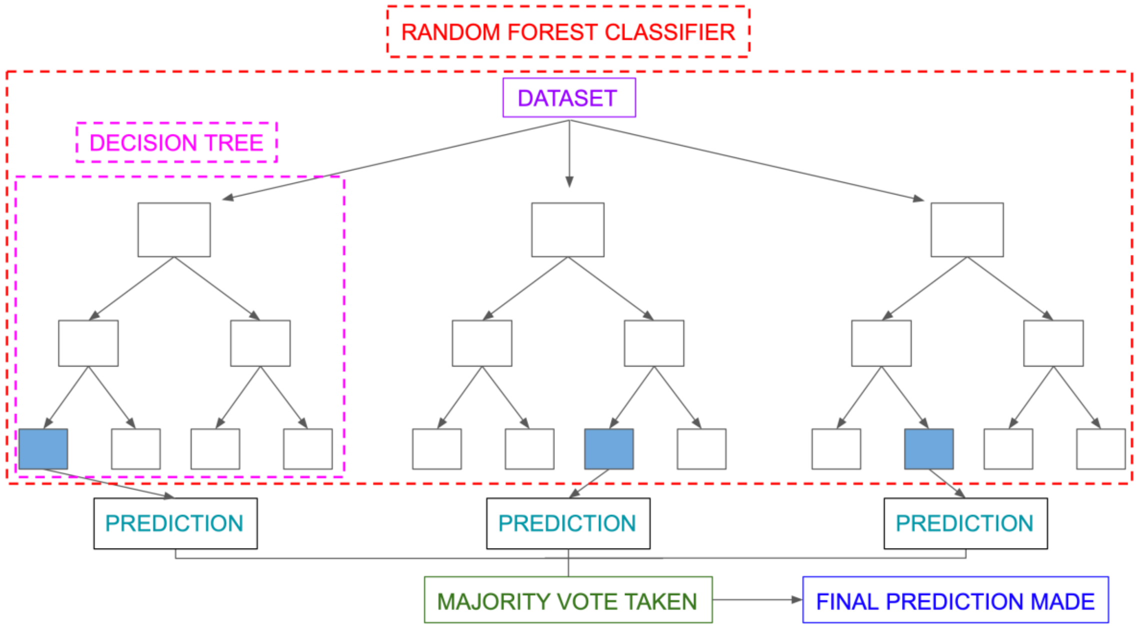
An Approach to ML and Security Onion

- New option from SecurityOnion posted 2 days ago
  - LogScan
  - Provided with Security Onion version 2.3
  - Only focuses on detecting anomalies in logs
  - Requires the vCPU to have AVX support (AVX2 recommended for better performance)
  - Not currently compatible with air gapped installations
  - Has the following model options:
    - K1: Searches for high numbers of login attempts from single IPs in a 1 minute window
    - K5: Searches for high ratios of login failures from single IPs in a 5 minute window
    - K60: Searches for abnormal patters of login failures from all IPs seen within a 1 hour window
  - It is log agnostic (interoperable among different types) but the current implementation only scans from built-in auth provider Kratos (identity management server)
  - Possible option to run alongside a custom ML implementation

Another Option: Binary Classification & Anomaly Detection (next slide)

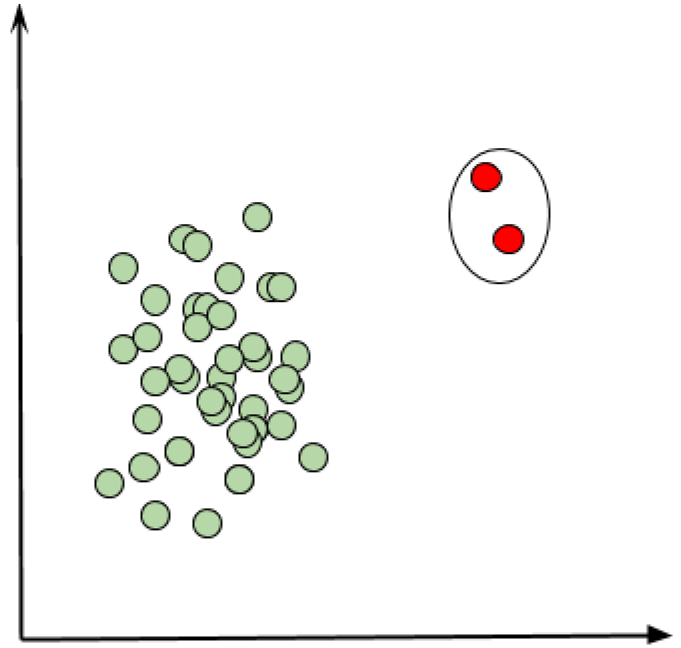
- Binary classification
  - Given a population of malicious and normal logs – finding the best function that separates them into two classes
  - Could be a line could be some other delineation
  - The function that does reasonably well at separating the two classes is the binary classifier
  - Random Forest can be used in a binary classification scenario
  - Classify known threats with high accuracy
  - Python libraries easiest to implement this with
    - Scikit learn
    - Pandas
    - Numpy

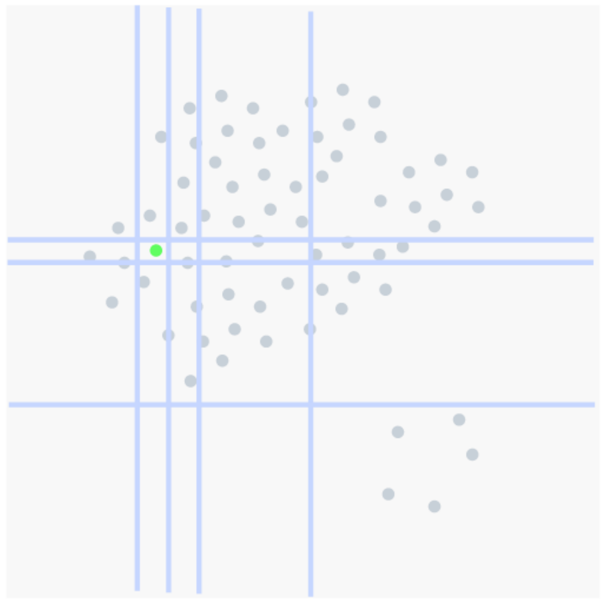




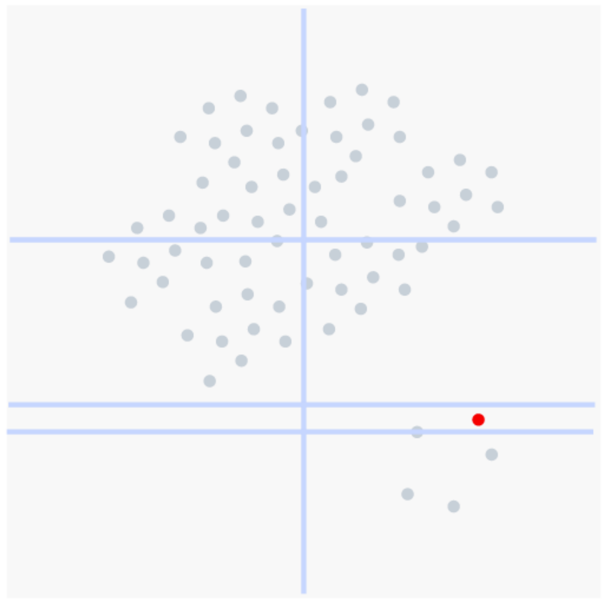
Anomaly Detection – Isolation Forests

- Can be utilized to identify anomalies (zero day)
- Selects a random dimension and randomly splits the data along that dimension
- The resulting subspaces are each their own sub-trees
- Process is repeated until every leaf of the tree represents a single data point from the dataset
- Isolation of a normal point takes more splits to single out which makes Isolation forests much more usable when identifying outliers



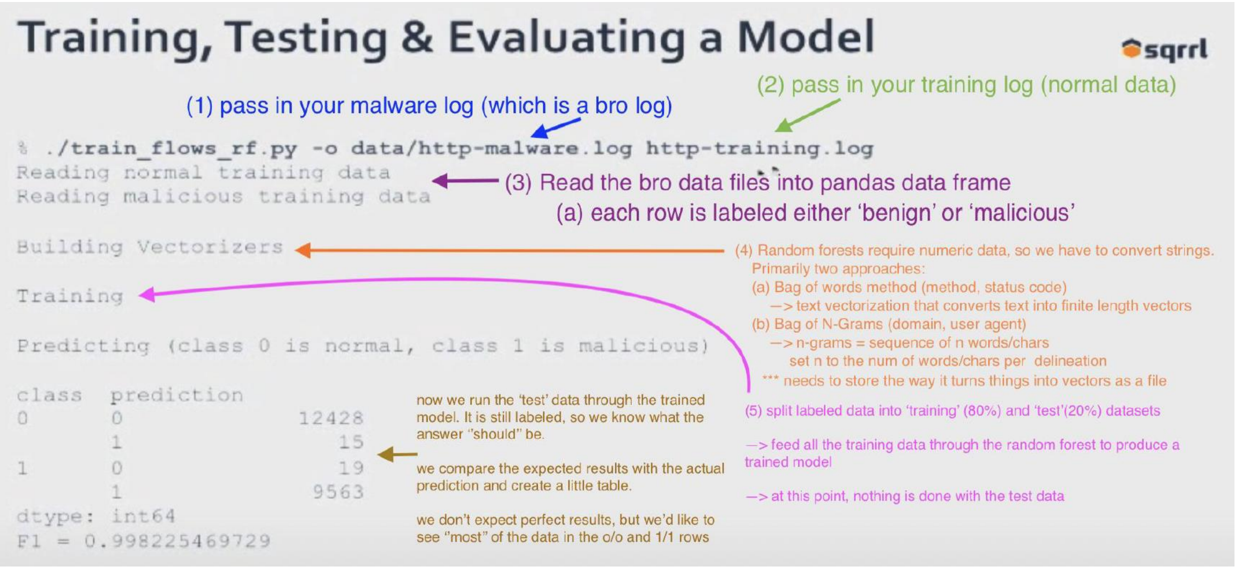


Isolation of a normal point



Isolation of an anomaly

Rough Outline of Implementation Plan



Training, Testing & Evaluating a Model

sqrrl

(1) pass in your malware log (which is a bro log)

(2) pass in your training log (normal data)

```
% ./train_flows_rf.py -o data/http-malware.log http-training.log
Reading normal training data
Reading malicious training data

Building Vectorizers

Training

Predicting (class 0 is normal, class 1 is malicious)

class  prediction
0      0           12428
       1              15
1      0              19
       1            9563
dtype: int64
F1 = 0.998225469729
```

(3) Read the bro data files into pandas data frame
(a) each row is labeled either 'benign' or 'malicious'

(4) Random forests require numeric data, so we have to convert strings.
Primarily two approaches:
(a) Bag of words method (method, status code)
—> text vectorization that converts text into finite length vectors
(b) Bag of N-Grams (domain, user agent)
—> n-grams = sequence of n words/chars
set n to the num of words/chars per delineation
*** needs to store the way it turns things into vectors as a file

(5) split labeled data into 'training' (80%) and 'test'(20%) datasets

—> feed all the training data through the random forest to produce a trained model

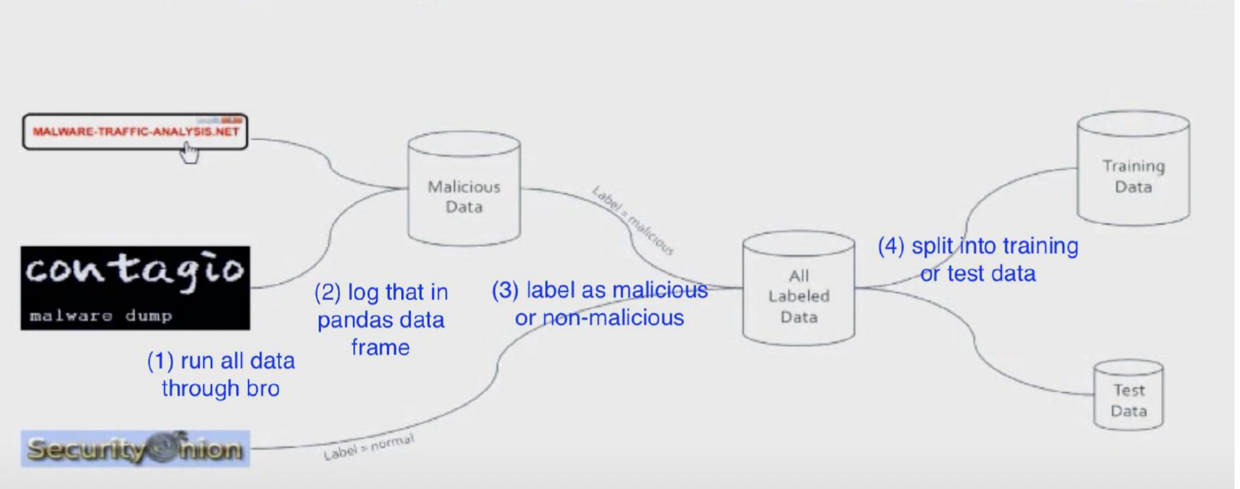
—> at this point, nothing is done with the test data

now we run the 'test' data through the trained model. It is still labeled, so we know what the answer "should" be.

we compare the expected results with the actual prediction and create a little table.

we don't expect perfect results, but we'd like to see "most" of the data in the o/o and 1/1 rows

Identifying Training & Test Data


sqrrl
MALWARE-TRAFFIC-ANALYSIS.NET
contagio malware dump
Security Onion
(1) run all data through bro
Malicious Data
(2) log that in pandas data frame
Label = malicious
Label = normal
(3) label as malicious or non-malicious
All Labeled Data
(4) split into training or test data
Training Data
Test Data

```
# install scikit-learn for machine learning
# install pandas for data manipulation
# install numpy for numerical operations

pip install scikit-learn pandas numpy

--------------------------------------------

import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split

# Create a Pandas datafram from a Bro log
bro_df = LogToDataFrame('/path/to/dns.log')

# clean the data accordingly

# split the data into training and test sets
# commonly 80% training and 20% test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# train the random forest model

from sklearn.ensemble import RandomForestClassifier
rf = RandomForestClassifier(random_state = 42)
rf.fit(X_train, y_train)

# Evaluate the Model

from sklearn.metrics import classification_report, accuracy_score
y_pred = rf.predict(X_test)
print(classification_report(y_test, y_pred))
print("Accuracy:", accuracy_score(y_test, y_pred))

# determine the most influential features (individual atributes)
# could include things like source/dest IP, source/dest port, protocol, packet size, etc

feature_importances = pd.DataFrame(rf.feature_importances_, index=X_train.columns,
columns=['importance']).sort_values('importance', ascending=False)
print(feature_importances)

#isolation forest for anomaly detection

from sklearn.ensemble import IsolationForest
iso_f = IsolationForest(random_state=42)
iso_f.fit(X_train)
anomalies = iso_f.predict(X_test) # will return -1 for anomalies
```

X - independent features
y - dependent features

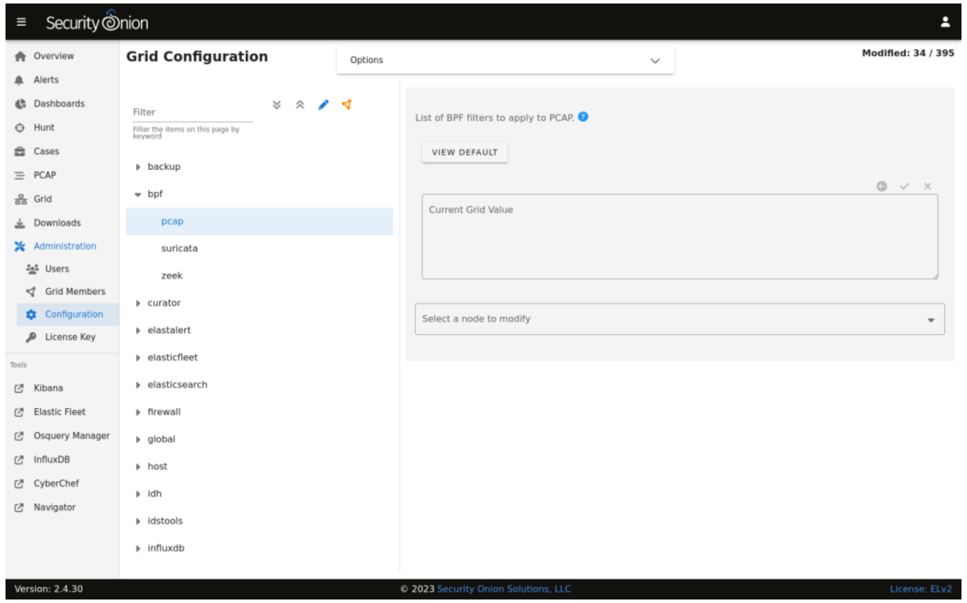
need both independent and dependent for training and test sets

Training set - trains the machine learning model

Testing set - evaulates the performance of the model

Security Onion Rule Tuning

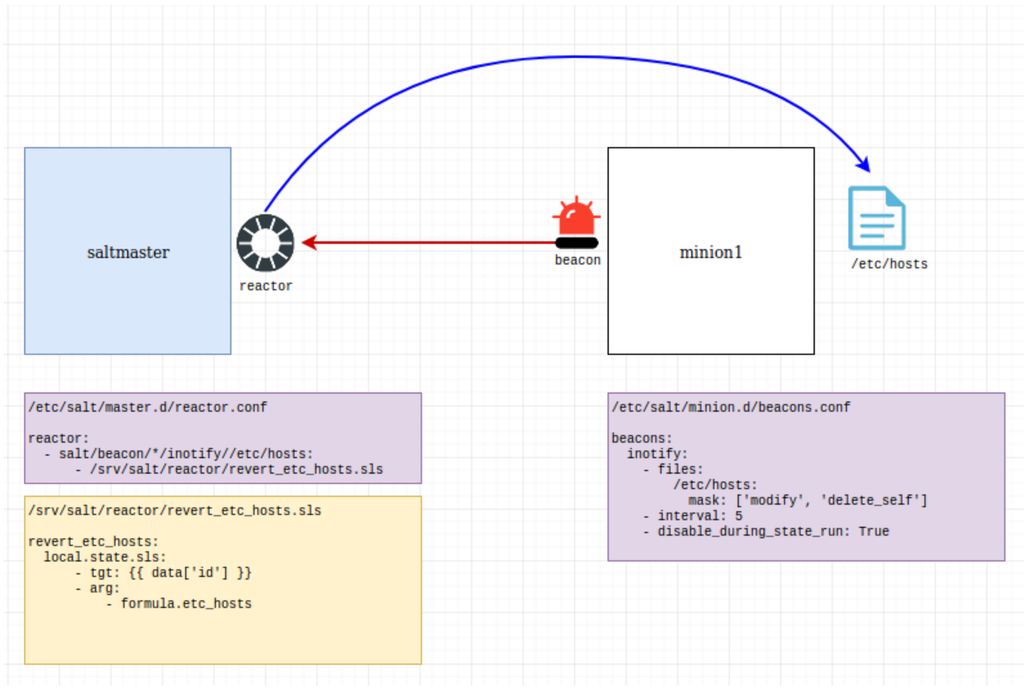
- Berkeley Packet Filter is a filtering system for the logs and packet data that is coming through the SIEM.
- The BPF filtering can be applied to Suricata and Zeek for logs as well as Stenographer for PCAP data.
- Suricata is the software that is used for alerting, so filters applied would be used to adjust alerting.
- The tuning can all be done from the SOC on the manager node, and to start with BPF ignoring all traffic and then including traffic to process.



List of BPF filters to apply to PCAP.

Salt

- Salt is the system that manages all processes on all nodes in Security Onion.
- Salt allows for code to be remotely executable on minion nodes.
- Beacons are a tool in Salt that are used for monitoring, they can monitor when a file is accessed or changed, shell activity, and network.
- Beacons can also be used to trigger Reactors, which are another system in Salt that can be used to execute code in response to a Beacon.



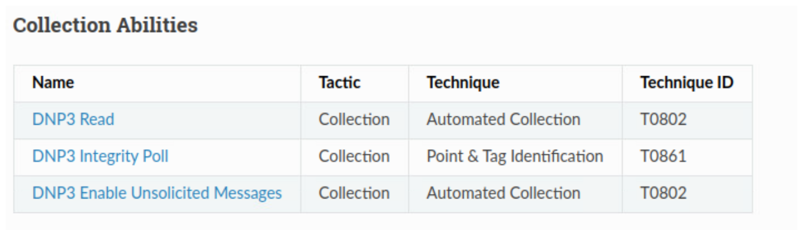
```
/etc/salt/master.d/reactor.conf

reactor:
  - salt/beacon/*/inotify//etc/hosts:
      - /srv/salt/reactor/revert_etc_hosts.sls
```

```
/srv/salt/reactor/revert_etc_hosts.sls

revert_etc_hosts:
  local.state.sls:
      - tgt: {{ data['id'] }}
      - arg:
          - formula.etc_hosts
```

```
/etc/salt/minion.d/beacons.conf

beacons:
  inotify:
    - files:
        /etc/hosts:
          mask: ['modify', 'delete_self']
    - interval: 5
    - disable_during_state_run: True
```

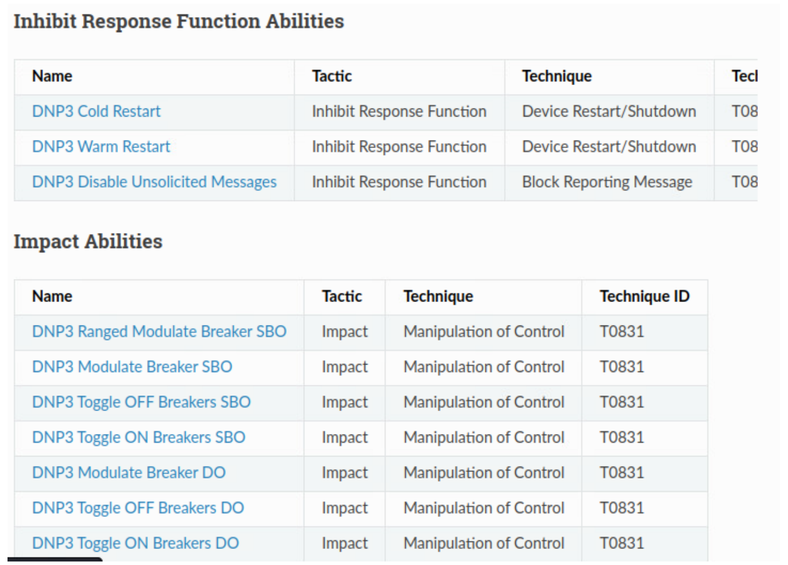
Mitre Caldera

1. Deploy Caldera agent
   - Need to run a series of commands on victim machine (Substation RTU or SIEM Sensor?)
2. Choose Adversary Profile
   - Tactic, technique, Ability



Collection Abilities

Name	Tactic	Technique	Technique ID
DNP3 Read	Collection	Automated Collection	T0802
DNP3 Integrity Poll	Collection	Point & Tag Identification	T0861
DNP3 Enable Unsolicited Messages	Collection	Automated Collection	T0802

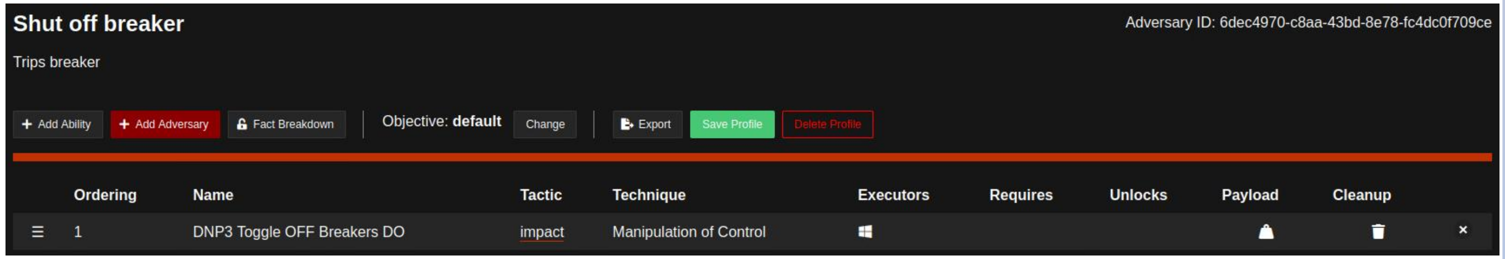


Inhibit Response Function Abilities

Name	Tactic	Technique	Tecl
DNP3 Cold Restart	Inhibit Response Function	Device Restart/Shutdown	T08
DNP3 Warm Restart	Inhibit Response Function	Device Restart/Shutdown	T08
DNP3 Disable Unsolicited Messages	Inhibit Response Function	Block Reporting Message	T08

Impact Abilities

Name	Tactic	Technique	Technique ID
DNP3 Ranged Modulate Breaker SBO	Impact	Manipulation of Control	T0831
DNP3 Modulate Breaker SBO	Impact	Manipulation of Control	T0831
DNP3 Toggle OFF Breakers SBO	Impact	Manipulation of Control	T0831
DNP3 Toggle ON Breakers SBO	Impact	Manipulation of Control	T0831
DNP3 Modulate Breaker DO	Impact	Manipulation of Control	T0831
DNP3 Toggle OFF Breakers DO	Impact	Manipulation of Control	T0831
DNP3 Toggle ON Breakers DO	Impact	Manipulation of Control	T0831



Shut off breaker

Adversary ID: 6dec4970-c8aa-43bd-8e78-fc4dc0f709ce

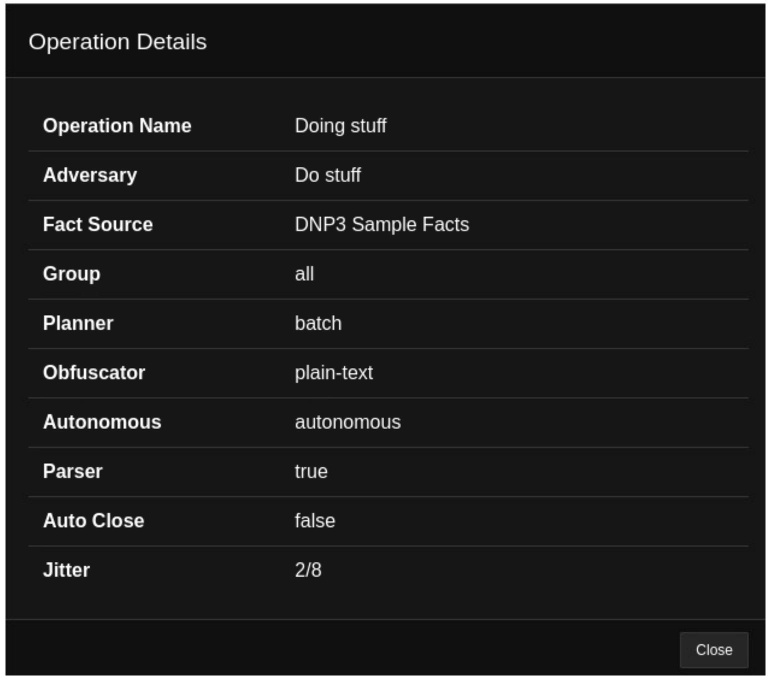
Trips breaker

Objective: **default**

	Ordering	Name	Tactic	Technique	Executors	Requires	Unlocks	Payload	Cleanup	
☰	1	DNP3 Toggle OFF Breakers DO	impact	Manipulation of Control	⊞			●	🗑	×

Mitre Caldera

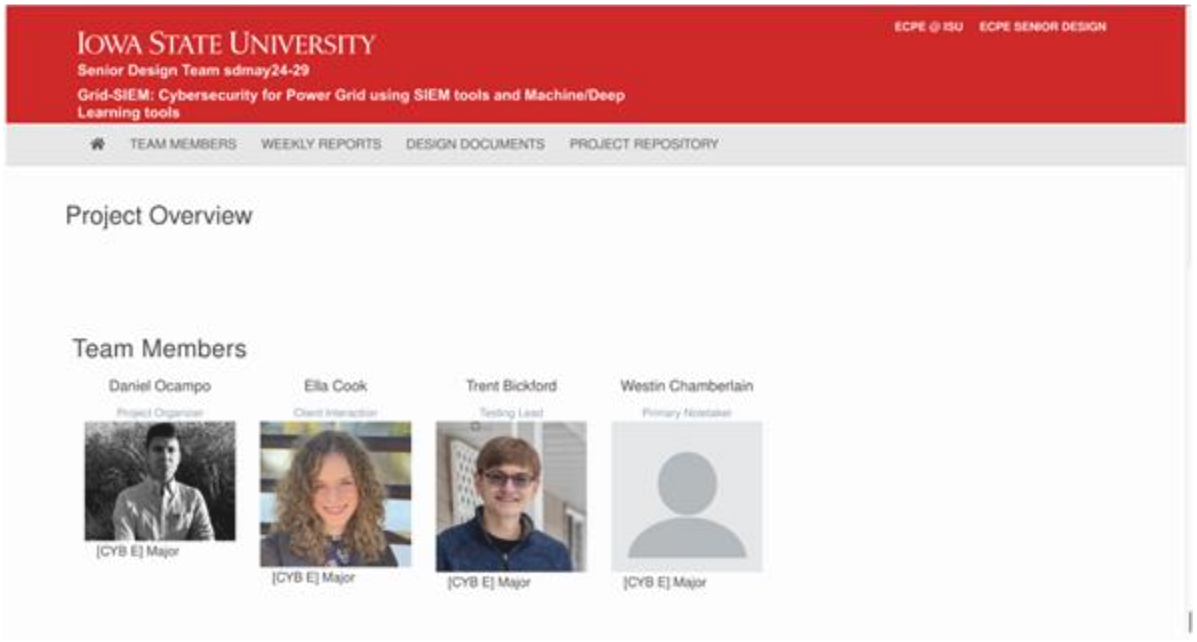
- Operation
  - Select new adversary, select "fact source"
  - Review operation after


Operation Details

Operation Name	Doing stuff
Adversary	Do stuff
Fact Source	DNP3 Sample Facts
Group	all
Planner	batch
Obfuscator	plain-text
Autonomous	autonomous
Parser	true
Auto Close	false
Jitter	2/8

End of Semester Checklist

- Submit our final draft for the engineering design document.
- Update the team website with our latest work. Pictures, bios, reports.
- Prepare official slide deck for the upcoming presentation.



IOWA STATE UNIVERSITY

Senior Design Team sdmay24-29

Grid-SIEM: Cybersecurity for Power Grid using SIEM tools and Machine/Deep Learning tools

Project Overview

Team Members

Daniel Ocampo
Project Organizer
[CYB E] Major

Ella Cook
Client Interaction
[CYB E] Major

Trent Bickford
Testing Lead
[CYB E] Major

Westin Chamberlain
Primary Notetaker
[CYB E] Major

https://sdmay24-29.sd.ece.iastate.edu/