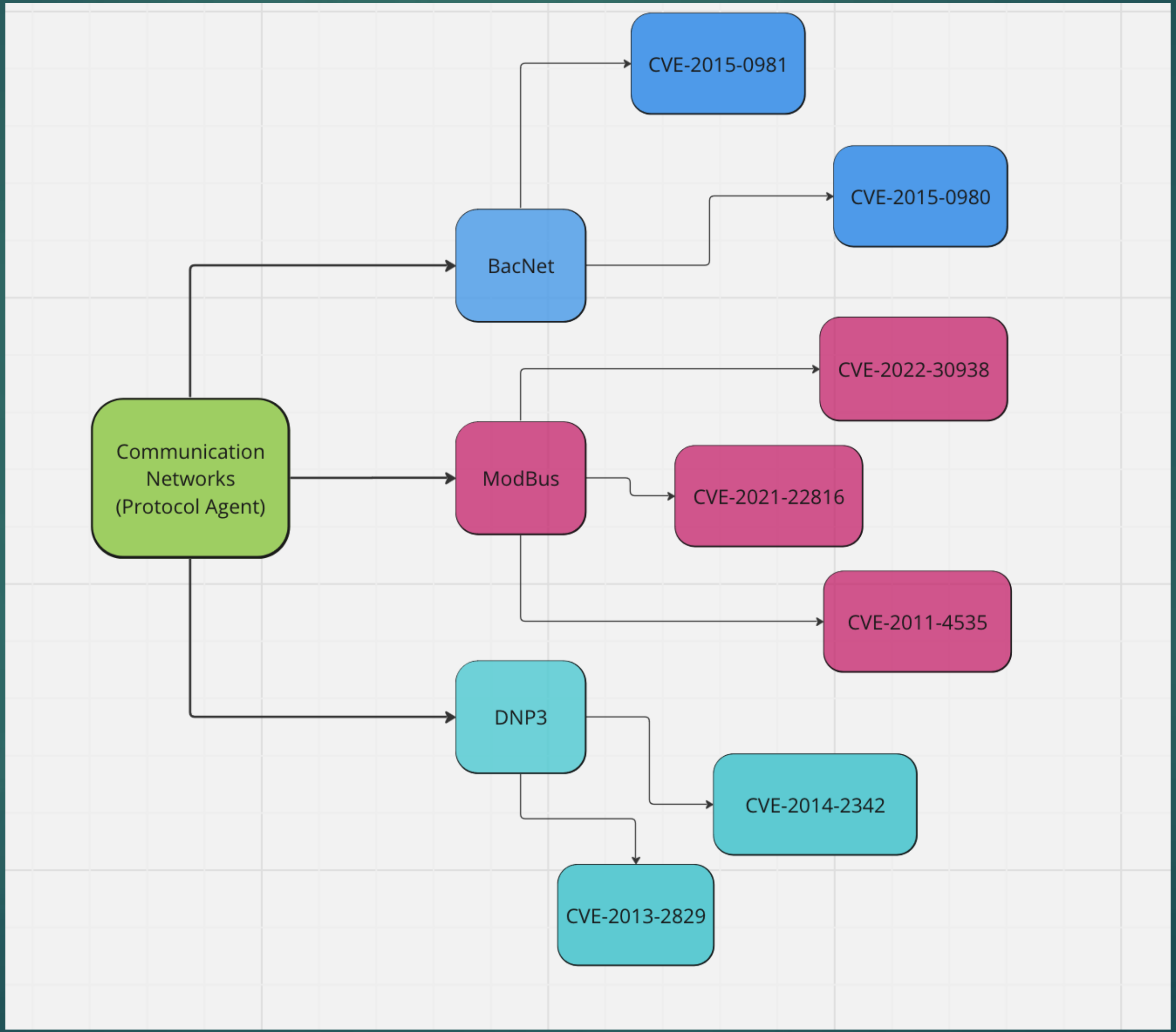
Grid-SIEM

GROUP 29 – SPRING SEMESTER

Attack Surface

- Protocols/Communication Networks
  - DNP3, ModBus, BacNet. Any other open ports.
- Control Systems
  - SCADA
- Data Storage and Processing Systems
  - Database of PowerCyber event logs. Or any other collection of data.
- Devices and Sensors
  - Can include sensors that collect data on electricity usage, equipment performance and other status conditions.
- Access Points and Interfaces
  - Gateways or routers?

Protocols



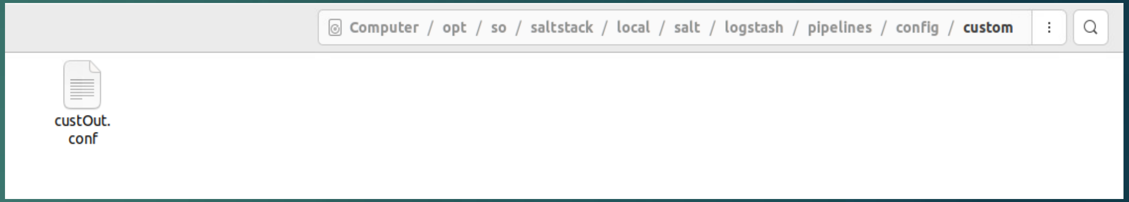
Communication Networks (Protocol Agent)
BacNet
CVE-2015-0981
CVE-2015-0980
ModBus
CVE-2022-30938
CVE-2021-22816
CVE-2011-4535
DNP3
CVE-2014-2342
CVE-2013-2829

Security Onion Task Division

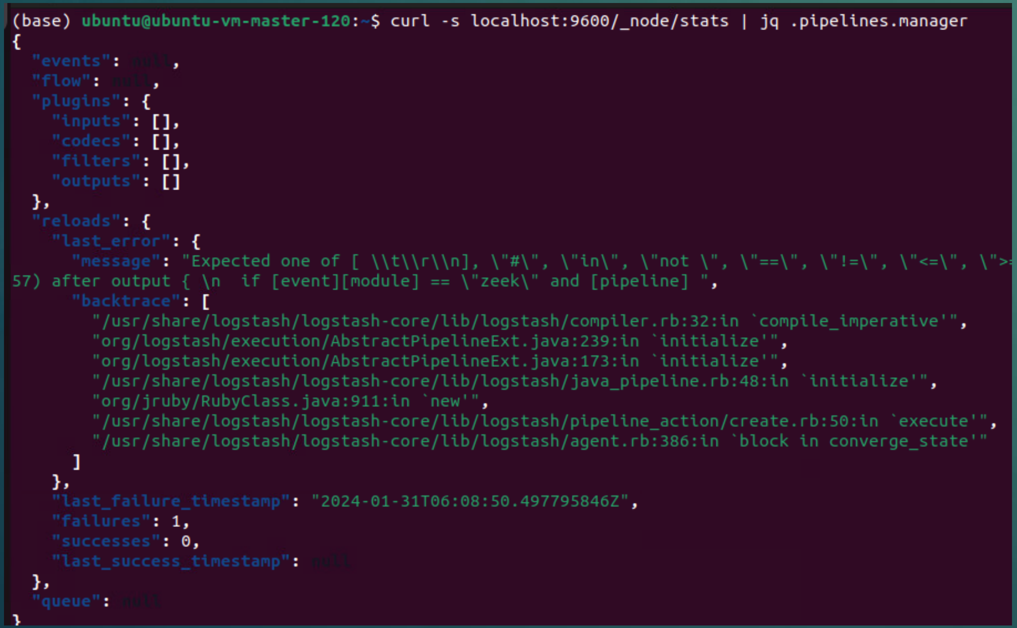
- Setting up nodes
  - Manager Search nodes need to be created for zones 1 and 3
  - Sensor nodes need to be created for zones 1 and 3
  - New nodes must request to become grid members.
- Configuring current node system
  - Need to forward Zeek logs into a file that can be accessed by ML
  - Adjust rules for alerting
  - Research Salt and how it can be used for code execution in responses
- Applications
  - Research more applications that would benefit the project

Security Onion Work

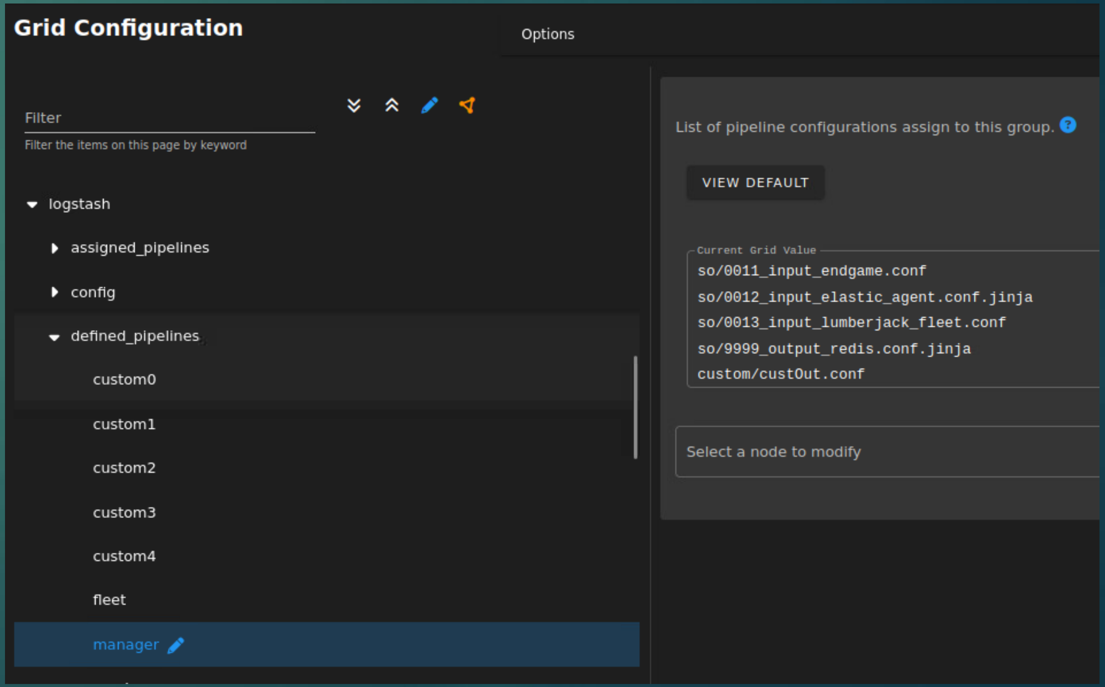
- Tried to setup a file location for Zeek logs on the Manager, creating a rule to forward the logs, but was not reading the logs



Computer / opt / so / saltstack / local / salt / logstash / pipelines / config / custom



```
(base) ubuntu@ubuntu-vm-master-120:~$ curl -s localhost:9600/_node/stats | jq .pipelines.manager
{
  "events": null,
  "flow": null,
  "plugins": {
    "inputs": [],
    "codecs": [],
    "filters": [],
    "outputs": []
  },
  "reloads": {
    "last_error": {
      "message": "Expected one of [ \\t\\r\\n], \"#\", \"in\", \"not \", \"==\", \"!=\", \"<=\", \">
57) after output { \n  if [event][module] == \"zeek\" and [pipeline] ",
      "backtrace": [
        "/usr/share/logstash/logstash-core/lib/logstash/compiler.rb:32:in `compile_imperative'",
        "org/logstash/execution/AbstractPipelineExt.java:239:in `initialize'",
        "org/logstash/execution/AbstractPipelineExt.java:173:in `initialize'",
        "/usr/share/logstash/logstash-core/lib/logstash/java_pipeline.rb:48:in `initialize'",
        "org/jruby/RubyClass.java:911:in `new'",
        "/usr/share/logstash/logstash-core/lib/logstash/pipeline_action/create.rb:50:in `execute'",
        "/usr/share/logstash/logstash-core/lib/logstash/agent.rb:386:in `block in converge_state'"
      ]
    },
    "last_failure_timestamp": "2024-01-31T06:08:50.497795846Z",
    "failures": 1,
    "successes": 0,
    "last_success_timestamp": null
  },
  "queue": null
}
```



Grid Configuration

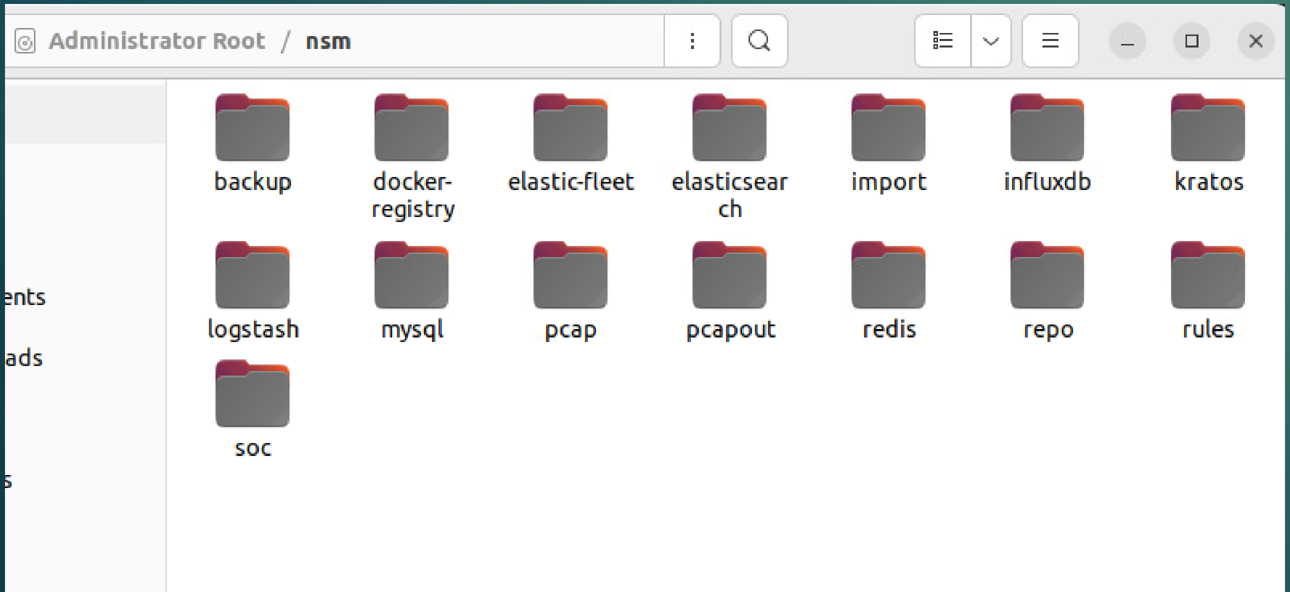
List of pipeline configurations assign to this group.

Current Grid Value

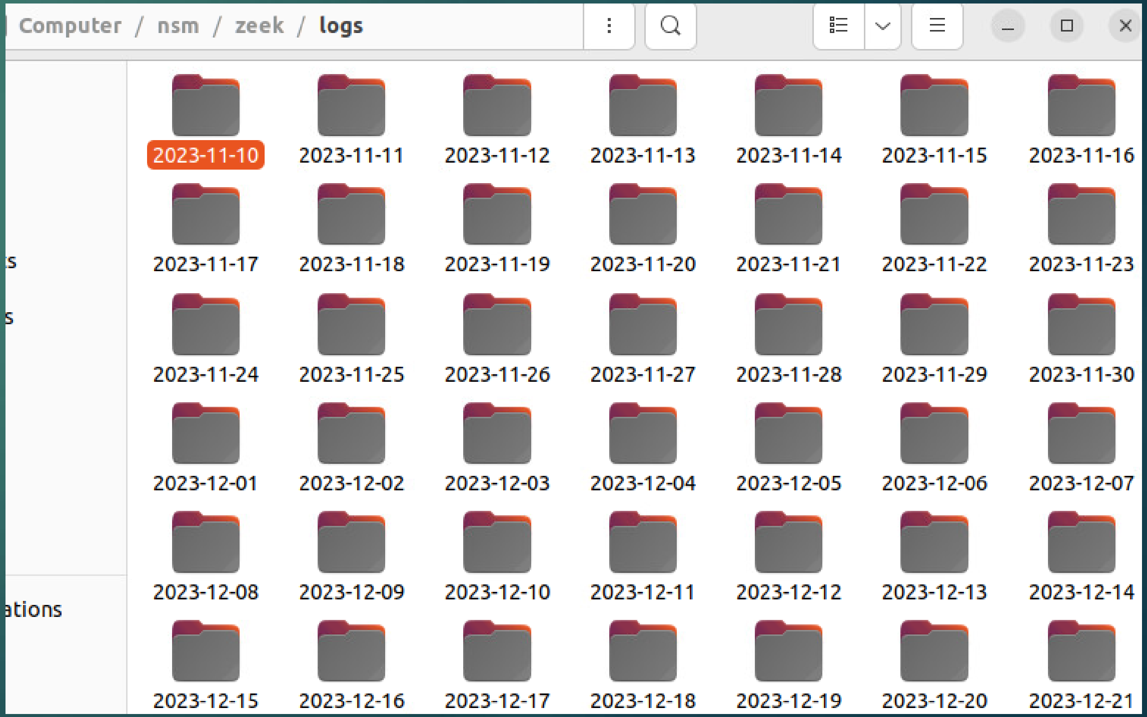
```
so/0011_input_endgame.conf
so/0012_input_elastic_agent.conf.jinja
so/0013_input_lumberjack_fleet.conf
so/9999_output_redis.conf.jinja
custom/custOut.conf
```

Security Onion work

- Zeek logs were being collected in the sensor, but not transferred to master files
- This week we will look at Security Onion in our team internal meeting to troubleshoot

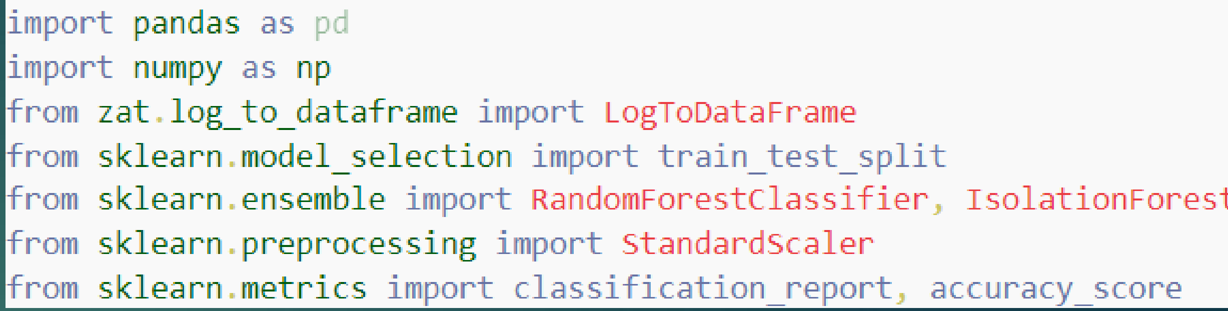


backup docker-registry elastic-fleet elasticsearch import influxdb kratos
logstash mysql pcap pcapout redis repo rules
soc



2023-11-10 2023-11-11 2023-11-12 2023-11-13 2023-11-14 2023-11-15 2023-11-16
2023-11-17 2023-11-18 2023-11-19 2023-11-20 2023-11-21 2023-11-22 2023-11-23
2023-11-24 2023-11-25 2023-11-26 2023-11-27 2023-11-28 2023-11-29 2023-11-30
2023-12-01 2023-12-02 2023-12-03 2023-12-04 2023-12-05 2023-12-06 2023-12-07
2023-12-08 2023-12-09 2023-12-10 2023-12-11 2023-12-12 2023-12-13 2023-12-14
2023-12-15 2023-12-16 2023-12-17 2023-12-18 2023-12-19 2023-12-20 2023-12-21

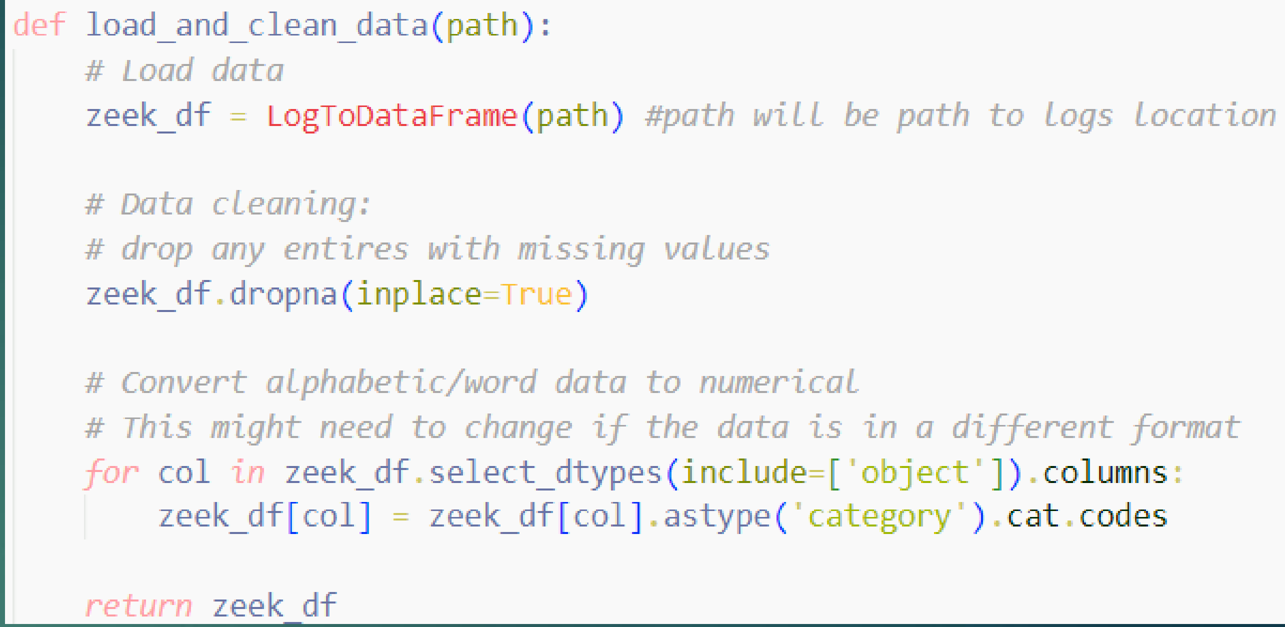
ML Work



```
import pandas as pd
import numpy as np
from zat.log_to_dataframe import LogToDataFrame
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier, IsolationForest
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report, accuracy_score
```

- Draft of functions implemented based on plan from last semester
- Imports:
  - Zat – Zeek analysis tools, provides the functionality to read zeek logs and convert them to pandas data frames
  - Pandas – data manipulation and analysis
  - Numpy – used for multi-dimensional arrays
  - Scikit-learn
    - To split the data for training and testing
    - To use the most efficient random forest and isolation forest algorithm implementations
    - Scaling -> ensures that all the features contribute equally to the final prediction and that features that occur more don't disproportionately influence the final prediction
    - Classification report – gives a overview of the performance of the ML model based on things like precision and recall (accuracy)
    - Accuracy score – percentage of correctly predicted outcomes

ML Work



```
def load_and_clean_data(path):
    # Load data
    zeek_df = LogToDataFrame(path) #path will be path to logs location

    # Data cleaning:
    # drop any entires with missing values
    zeek_df.dropna(inplace=True)

    # Convert alphabetic/word data to numerical
    # This might need to change if the data is in a different format
    for col in zeek_df.select_dtypes(include=['object']).columns:
        zeek_df[col] = zeek_df[col].astype('category').cat.codes

    return zeek_df
```

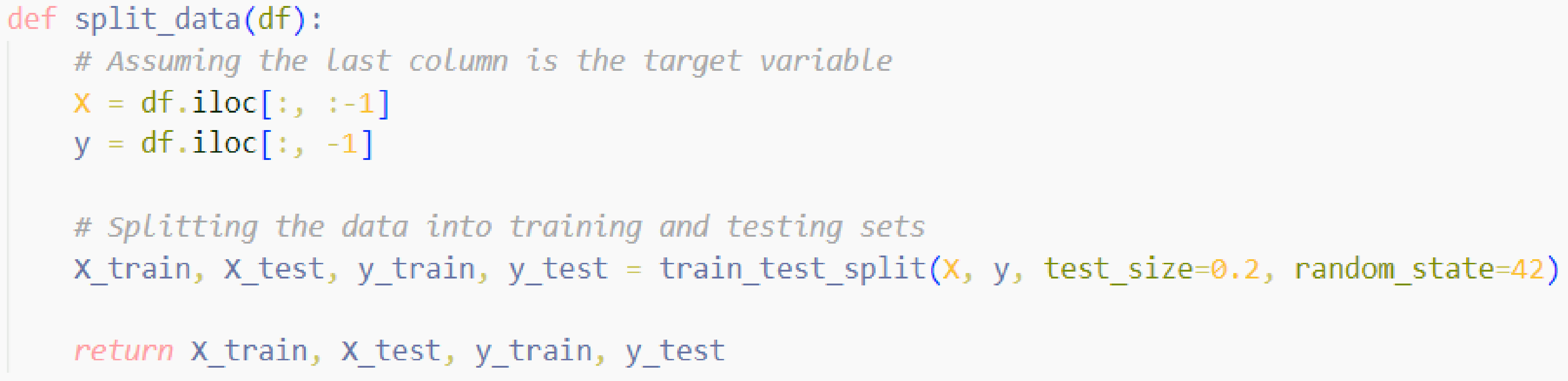
- Cleaning function:
  - The data is loaded using the

    LogToDataFrame function from ZAT, where

    Path is the path directory the logs are stored
  - The zeek_df.dropna(inplace=true) is used to drop any data entries in the log that contain any missing info
  - The for loop – loops over all the columns in the df that contain data then it converts each word/alphabetic data type to a unique numerical value.
    - Example: it would map each unique string with a numerical value so that it can be processed by the ML models
      - An entry or red green red might be 0 1 0 if red=0 and green=1

ML Work

- Function to split training and testing data
  - The line x = df.iloc[:, :-1] creates variable x that contains all the columns of the data frame except the last one – this is a standard used for ML
  - The line y = df.iloc[:, -1] creates variable y that is only the last column of the data frame – this is used as a target variable in the ML
  - The line with the train_test_split is also standard with the test_size = 0.2 meaning that 20% of the data will be used as test data and 80% will be used as training data



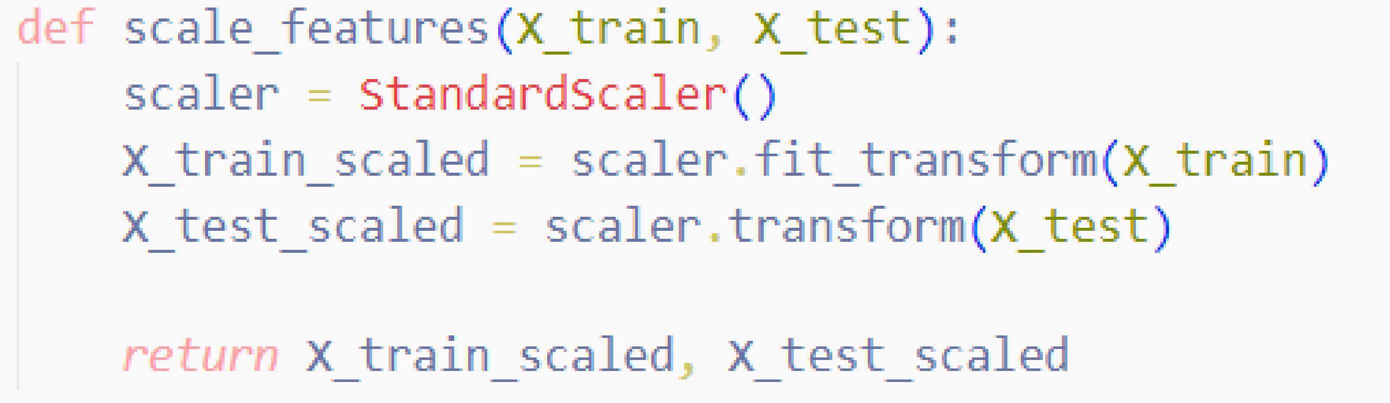
```python
def split_data(df):
    # Assuming the last column is the target variable
    X = df.iloc[:, :-1]
    y = df.iloc[:, -1]

    # Splitting the data into training and testing sets
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

    return X_train, X_test, y_train, y_test
```

ML Work

- Scale function
  - The standard scaler is used to make sure that all the data has a common scale
  - The scaler.fit_transform and scaler.transform adjusts the data so that it is all consistent
  - The entire purpose of the scaler is to prepare the training and testing data for the machine learning models

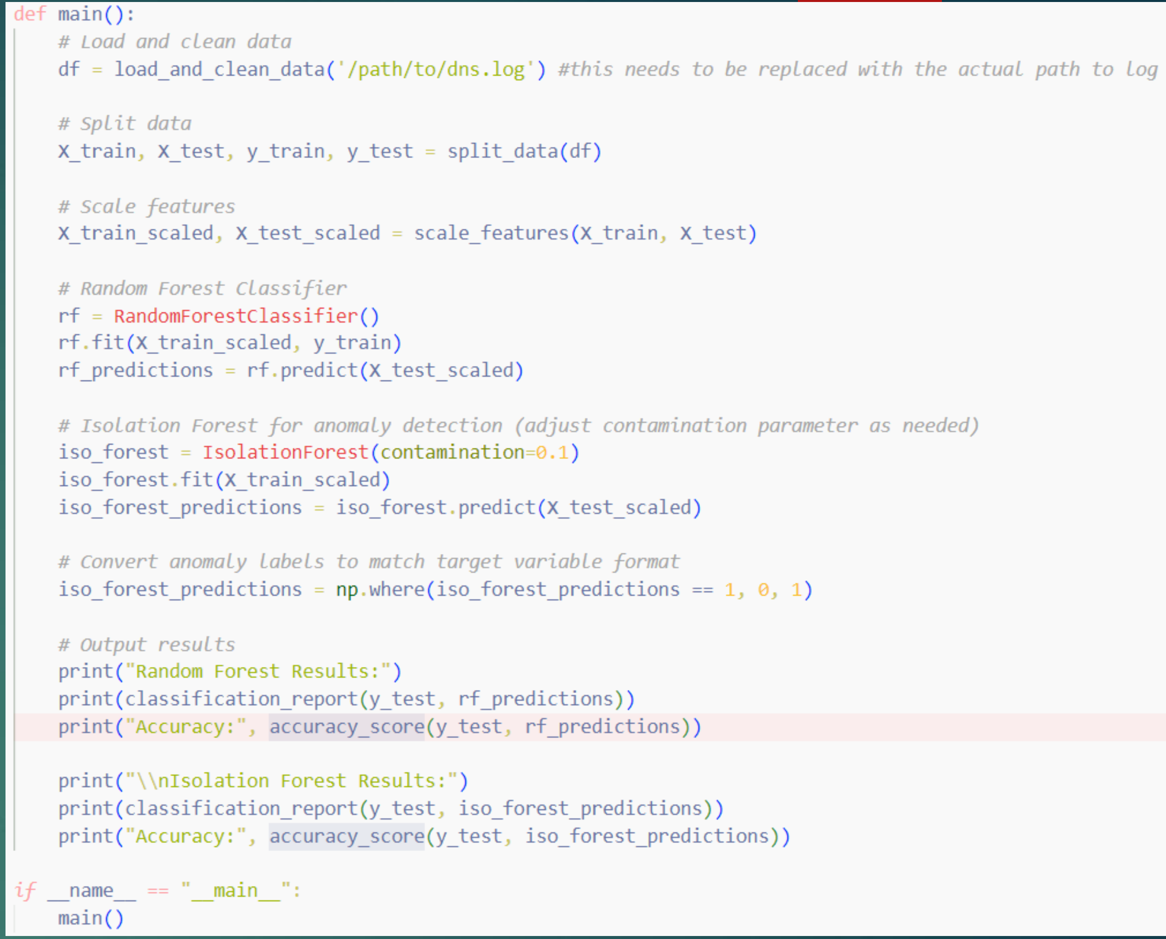


```
def scale_features(X_train, X_test):
    scaler = StandardScaler()
    X_train_scaled = scaler.fit_transform(X_train)
    X_test_scaled = scaler.transform(X_test)

    return X_train_scaled, X_test_scaled
```

ML Work

- Main
  - Loads and cleans the data from the log dir location
  - Split data into train (80%) and test (20%)
  - Scales the train and test data
  - Creates an instance of Random Forest from scikit learn library then fits the scaled train and test data
  - Makes a prediction using the same random forest implementation from scikit learn
  - Creates an instance of isolation forest from scikit-learn, teaches the I.F. with training data, and produces a prediction
  - The np.where function for the I.F. makes the results from the I.F. in a normal format
  - Finally the random and isolation forest results are output and the accuracy for each is provided as well



```
def main():
    # Load and clean data
    df = load_and_clean_data('/path/to/dns.log') #this needs to be replaced with the actual path to log

    # Split data
    X_train, X_test, y_train, y_test = split_data(df)

    # Scale features
    X_train_scaled, X_test_scaled = scale_features(X_train, X_test)

    # Random Forest Classifier
    rf = RandomForestClassifier()
    rf.fit(X_train_scaled, y_train)
    rf_predictions = rf.predict(X_test_scaled)

    # Isolation Forest for anomaly detection (adjust contamination parameter as needed)
    iso_forest = IsolationForest(contamination=0.1)
    iso_forest.fit(X_train_scaled)
    iso_forest_predictions = iso_forest.predict(X_test_scaled)

    # Convert anomaly labels to match target variable format
    iso_forest_predictions = np.where(iso_forest_predictions == 1, 0, 1)

    # Output results
    print("Random Forest Results:")
    print(classification_report(y_test, rf_predictions))
    print("Accuracy:", accuracy_score(y_test, rf_predictions))

    print("\\nIsolation Forest Results:")
    print(classification_report(y_test, iso_forest_predictions))
    print("Accuracy:", accuracy_score(y_test, iso_forest_predictions))

if __name__ == "__main__":
    main()
```

Past Senior Design Projects

- Simulating Cyberattacks on a Power Grid to Determine Potential Impacts - sdmay23-02
  - This was useful because they used Pandas to simulate a false data injection which is like the data or function injection that we are using in our project.
- CPS-CDC PowerCyber Testbed (sddec14-07)
  - Could provide some insight to the inner workings of PowerCyber

https://seniord.ece.iastate.edu/projects/