Anomaly Detection using Machine Learning

Report by Manarth Bhavsar

Agenda

- Overview
- Background
- Problem Statement
- Data Collection & Preprocessing
- Methodology
- Results
- Challenges & Areas of Improvement
- Conclusion
- References

Leveraging ML Technique for Cybersecurity

Overview

The main purpose of this project is use Machine Learning in cybersecurity to enhance threat detection, incident response, and overall resilience against cyber attacks. By leveraging advanced ML algorithm, the project seeks to develop automated solutions for identifying, analyzing, and mitigating security threats across various digital environments.

Background

Cybersecurity has become a critical concern in today's digital age, with cyber attacks posing significant risks to businesses, governments, and individuals alike. The ever-evolving nature of these threats demands advanced solutions that can adapt and respond effectively. Machine learning has emerged as a promising approach to bolster cybersecurity defenses by leveraging data-driven insights to detect and prevent cyber threats in real-time.

Problem Statement

With the increasing complexity and sophistication of cyber threats, traditional cybersecurity measures are struggling to keep pace. Organizations need innovative solutions to proactively detect and respond to cyber attacks before they cause significant damage. This project aims to leverage machine learning technique to enhance cybersecurity by developing an anomaly detection system capable of identifying and mitigating potential security threats in real-time.

Data Collection & Preprocessing

- Integrated Splunk to collect Linux System logs in real time.
- Extracted features including Time stamp, Process ID, Event Template, etc into columns.
- Processed log data converted to CSV for integration with ML model.
- For Example, took sample data from Log-Hub Repository, a curated collection of diverse log datasets.

Isolation Forest

Mechanism:

- Designed specifically for Anomaly Detection
- Identifies outliers by isolating data points using random splits.
- Randomly selects and splits features into minimum and maximum values to obtain the outliers from the dataset by the use of multiple trees.

Comparison with Random Forest:

- Generally useful for large-scale anomaly detection in compare to Random Forest.

One Hot Encoder

Reason to use One Hot Encoder:

- Isolation forest only accepts numerical values so to convert the categorial features into 1’s & 0’s, we use one hot encoder.

Comparison with Word2Vec:

- OHE is beneficial for large number of categories and Word2Vec works best with large vocabulary in dataset.
- Word2Vec requires training on heavy dataset whereas OHE doesn’t require any training.

Methodology

```python
import splunklib.client as client
import csv


# Connect to Splunk
service = client.connect(host='192.168.1.187', port='8089', username='admin', password='password')

# Define search query
search_query = 'source="/var/log/syslog" host="enpm685" sourcetype="syslog" | search *'

# Run search
job = service.jobs.create(search_query)

# Wait for search to finish
while True:
    while not job.is_ready():
        pass
    if job['isDone'] == '1':
        break

# Get results
results = job.results()

# Process results and save to CSV
with open('output.csv', 'w', newline='') as csvfile:
    csvwriter = csv.writer(csvfile)
    csvwriter.writerow(['Month', 'Date', 'Time', 'Level', 'Component', 'PID', 'Content', 'EventId', 'EventTemplate'])  # Write header row
    for result in results:
        # Process result and extract fields as needed
        csvwriter.writerow([
            result['Month'],
            result['Date'],
```

```python
# Encode categorical variables using one-hot encoding

categorical_columns = ['Month', 'Date', 'Time', 'Level', 'Component', 'PID', 'Content', 'EventId', 'EventTemplate']

# Perform one-hot encoding on categorical columns
onehot_encoder = OneHotEncoder(sparse_output=False)
encoded_columns = pd.DataFrame(onehot_encoder.fit_transform(logs[categorical_columns]))
encoded_columns.columns = onehot_encoder.get_feature_names_out(categorical_columns)


# Concatenate encoded columns with original DataFrame
logs_encoded = pd.concat([logs.drop(columns=categorical_columns), encoded_columns], axis=1)

# print("Encoded Columns:")
# print(encoded_columns)
```

- Connection established to Splunk on Ubuntu System to fetch raw data.
- Converted the raw data into feature based csv for further implementation.
- Used OneHotEncoder to convert categorial features into numerical values.
- Trained the Isolation Forest model for the dataset.
- Generated printable output with EventID & Event Template for instant action.
- Downloadable CSV file for further investigation.

Results

```
Anomaly detected at LineId 14.0, Event Templates: session opened for user <*> by (uid=<*>)
Anomaly detected at LineId 16.0, Event Templates: ALERT exited abnormally with [1]
Anomaly detected at LineId 33.0, Event Templates: check pass; user unknown
Anomaly detected at LineId 34.0, Event Templates: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*>
Anomaly detected at LineId 36.0, Event Templates: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*>
Anomaly detected at LineId 37.0, Event Templates: check pass; user unknown
Anomaly detected at LineId 38.0, Event Templates: authentication failure; logname= uid=0 euid=0 tty=NODEVssh ruser= rhost=<*>
Anomaly detected at LineId 73.0, Event Templates: session opened for user <*> by (uid=<*>)
Anomaly detected at LineId 74.0, Event Templates: session closed for user <*>
Anomaly detected at LineId 75.0, Event Templates: ALERT exited abnormally with [1]
Anomaly detected at LineId 76.0, Event Templates: session opened for user <*> by (uid=<*>)
Anomaly detected at LineId 80.0, Event Templates: ALERT exited abnormally with [1]
Anomaly detected at LineId 139.0, Event Templates: ALERT exited abnormally with [1]
Anomaly detected at LineId 142.0, Event Templates: session opened for user <*> by (uid=<*>)
Anomaly detected at LineId 147.0, Event Templates: ALERT exited abnormally with [1]
```

```
Anomaly detected at LineId 1871.0, Event Templates: connection from <*> (<*>) at <*>:<*>:<*>
Anomaly detected at LineId 1872.0, Event Templates: connection from <*> (<*>) at <*>:<*>:<*>
Anomaly detected at LineId 1873.0, Event Templates: connection from <*> (<*>) at <*>:<*>:<*>
Total number of anomalies: 100
Downloaded anomalies_output.csv to your local system
```

Challenges Faced

- Handling large amount of data real time needs high traffic networks & cloud-based systems.
- Working with imbalanced dataset where anomalies are more than outliers can cause problems.
- Resource Constraints such as memory, processing power, and storage capacity, which can restrict the deployment and operation of machine learning models in resource-constrained environments.

Room For Improvement

- Exploring advanced feature engineering techniques to capture more nuanced patterns in log data.
- Combining multiple models for improved performance.
- Continuously updating models with new data and retraining them to adapt to changing threat landscapes.

Summary

- Analyzed Linux system logs to identify anomalies and detect potential security threats.
- Integrated data from various sources, including Splunk, to collect relevant log data for analysis.
- Processed the collected data into a suitable format for training machine learning model.
- Utilized Isolation Forest to identify suspicious activities within the log data. Output the data in interpretable form for instant action as well as further investigation.

Conclusion

- Demonstrated the effectiveness of machine learning in bolstering cybersecurity defenses.
- Highlighted the capability of data-driven approaches to proactively identify and respond to security threats.
- Successfully showcased the ability of machine learning algorithms to detect anomalies and potential security breaches.
- Acknowledged ongoing challenges in data preprocessing complexity and the need for continuous innovation in cybersecurity practices.

References

Linux Dataset: https://github.com/logpai/loghub

https://blog.paperspace.com/anomaly-detection-isolation-forest/

https://dev.splunk.com/enterprise/docs/devtools/python/sdk-python/