Capstone Project: Anomaly Detection in Low-Band Resource-Constrained Networks

Maureen Van Devender, Chandler Armagost, Parker Arrington, Tristan Clark, Eric Sternberg
School of Computing, University of South Alabama

Introduction

The University of South Alabama and its School of Computing offers undergraduate degrees in Information Technology, Information Systems, and Computer Science. Graduates of these programs are required to participate in a senior capstone experience course (CIS-497) which gives opportunity for teams of 4-5 students to work on real-world projects proposed by industry, academic, and government partners to the school. The Center for Forensics, Information Technology, and Security (CFITS) coordinates the participation of various partners to propose projects that help students put into practice the systems development lifecycle methods they have learned as part of their curriculum.

Problem Domain

The United States develops and provides current and future global space and high-altitude capabilities to U.S. military forces, allies, and partners. Of these capabilities is the use of satellites for government and commercial communications. However, due to the operational nature of the satellites as they rotate around the earth, they are constrained to limited communication windows. General anomaly detection is typically not feasible due to this operational overhead and monitoring/reporting are typically limited to routine system health checks. Ground station teams must determine any anomalies in satellite operations or baseline signature -- such anomalies can be signals of a threat that must be addressed. Therefore, ground station operators need a way to consistently track satellite network and communication anomalies to effectively defend assets.

Objective and Goals

This scope of the project is:

- to analyze the problem
- research available technology that will begin to lay the groundwork for ground operators to have a way to automatically track and monitor network and communication anomalies
- develop a proof-of-concept model

The proposed solution should be a unique idea for the client to base a fully-fledged anomaly detection system around.

Functional Requirements

1. *The system shall allow for the ground operator to start the receiving of data from satellites to begin a cache of data for real-time analysis.*

   Plan of Action: Meeting this requirement is centralized around the cache system and startup. The startup will be completed automatically upon connection to the satellite network to allow for a simple workflow. The cache system will be implemented through a temp file that will be stored locally on the device.

2. *The system shall allow for the ground operator to start detecting anomalies within a satellite mesh network.*

   Plan of Action: A startup runtime script will start the anomaly detection software, look for network connections and begin anomaly detection upon connection. There will be a separate script that the operator can run if there is not network connections at startup.

3. *The system shall present the operator with any anomalous data that is present within the system.*

   Plan of Action: A machine learning algorithm for anomaly detection will be selected based on researching and testing available models. Because the client is unable to give actual data, sample data will be used. The goal is to achieve 90% accuracy in detecting anomalies. Anomalous activity will be reported to the operator in the form of an alert box on the system display.

4. *The system shall allow for the ground operator to restart the system, continue scanning for anomalous data if a false positive occurs or if no action can be taken at that time.*

   Plan of Action: Anomalous results will be displayed in a clear and easy-to-understand method. The user can decide whether the data is anomalous and whether anything must be done to respond.

Non-Functional Requirements

1. **System Constraints** - must be able to run on resource-constrained systems on a Linux-like operating system.
2. **Time** - the system should be able to run in near real-time to allow for the best attempt at stopping anomalous events before critical issues begin to occur.
3. **Accuracy** - the system should be able to correctly identify anomalous events at least 90% of the time.
4. **Usability** - this system should be able to run with little to no experience with the system. Most work will be done in the background and does not require user control.

Requirements Assessment

The first functional requirement was completed through the collection of packets using the PyShark [2] Python library, which scans packets and stores them in a cache that holds the packets and allows the system to analyze the data and search for anomalies. The system holds a small amount of data before analyzing it in sections to get the system as close to real-time as possible.

The second functional requirement was met by creating a Python script that can be run on the ground system and checks five-second intervals of packet traffic for anomalies.

The third functional requirement was met by exporting the IP layer data of any packets identified as anomalous data to a static file where the operator can inspect the data to determine whether it is anomalous and if any action needs to be taken.

The fourth functional requirement of allowing the operator to restart the system is not achieved in the timeframe allowed for this project. The system continues to scan data and does not stop until the communication system is shut down outside of the functions of this solution.

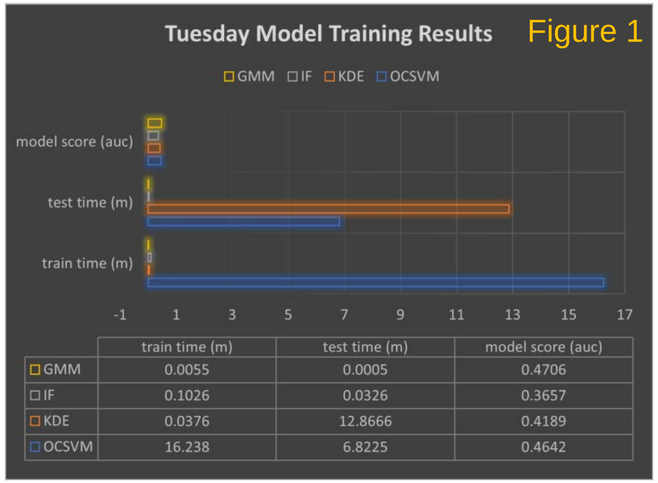

Figure 1

	train time (m)	test time (m)	model score (auc)
GMM	0.0055	0.0005	0.4706
IF	0.1026	0.0326	0.3657
KDE	0.0376	12.8666	0.4189
OCSVM	16.238	6.8225	0.4642

```
After splitting flows, the number of subflows: 37 and each of them has at least 2 packets.
'_pcap2flows()' ends at 2022-04-13 12:32:35 and takes 0.0011 mins.
'_flow2features()' starts at 2022-04-13 12:32:35
True
'_flow2features()' ends at 2022-04-13 12:32:35 and takes 0.0 mins.
Total Down Time between Intervals: 0.0003522000000000247
```

Figure 2

Test Results

Two machine learning packages were discovered through research and analyzed for fit. These were NetML[3] and PyOD [4]. NetML was determined to be the best fit due to it being defined as a network detection solution.

Once NetML was identified as the best fit, testing of the models was conducted using the testing system built-in to NetML. The process was tested by obtaining sample PCAP files[1]. The model was trained using malicious and normal PCAP files. The datasets used for training and testing were the Tuesday working hours data that included both normal data and brute force attacks [1]. The test results were inconclusive in determining the best way to train the models for highest accuracy. The Tuesday Model Training Results (Figure 1) shows an inverse correctness. As a result, the best model is the **Isolation Forest mode**l with an actual model score close to 0.64. This is well below the planned accuracy of .90 or higher.

The future plan for testing is to have more data to test with and to allow for more verbose capture of the PCAP for testing. This is not currently achieved due to hardware limitations available to the project team, particularly RAM.

Testing the ability of the system to capture live data in near realtime was done next. The capture was done using Pyshark 0.4.5 [2], a Python wrapper for tshark on a Kali Linux system. Figure 2 shows the downtime between five-second captures is approximately 0.00035 seconds. This was determined to be within reason since most connections and traffic within a network would last longer than this.

Bibliography

1. Brunswick, U. o. (2017). Intrusion Detection Evaluation Dataset (CIC-IDS2017). From https://www.unb.ca/cic/datasets/ids-2017.html
2. KimiNewt. (2022). PyPi. From Pyshark Documentation: https://pypi.org/project/pyshark/
3. Kun Yang, S. K. (2020). PyPi. From NetML Documentation: https://pypi.org/project/netml/
4. Yue Zhao, Z. N. (2019). Journal of Machine Learning Research. From PyOD: A Python Toolbox for Scalable Outlier Detection: https://pyod.readthedocs.io/en/latest/

QR Code