### Overview

* Objective: Build advanced visualization and monitoring tools to provide deep insights into network operations within Nutanix environments.
* Ideas:
  + Interactive Network Maps: Create interactive maps that visualize network topology, traffic flows, and performance metrics.
  + Anomaly Detection Dashboards: Develop dashboards that highlight network anomalies and potential issues using machine learning algorithms.

**Description:**

The goal of this project is to develop an AI-driven solution that creates interactive maps and dashboards that visualize network topology, traffic flows, and performance metrics, in order to detect anomalies in network traffic.

The project addresses concerns regarding network security traffic by identifying potential security threats or system malfunctions, such as DoS attacks, unauthorized access, and unusual network behavior. We aim to provide administrators with deeper visibility into real-time network activities, facilitating proactive issue detection and informed decision-making for troubleshooting and optimization.

**WHY:**

No matter how well-built systems can be, they are always susceptible to attacks. We decided to create a product that gives Nutanix customers a full-fledged view of their traffic, helping them detect and deter networking attacks.

### Attacks

**DOS:** make a computer or network unavailable to its intended users by overwhelming it with a flood of illegitimate requests or traffic

**Probing:** testing or checking a system to find weaknesses or vulnerabilities that could be exploited later

Example: Someone might send small bits of information to a computer or network to see how it responds, looking for ways to break in or cause harm.

**U2R:** someone with user-level access to a system tries to gain administrator or root-level access

Example: If someone hacks into a computer as a regular user and then tries to escalate their privileges to gain control over the entire system.

**R2L:** R2L attacks involve an outsider trying to gain unauthorized access to a local network or computer from a remote location.

Example: Someone outside the company network trying to exploit a vulnerability in a company server to gain access to sensitive information.

Relevant features:

* 1. DOS: “source bytes” and “percentage of packets with errors”
  2. Probing: “duration of connection” and “source bytes”
  3. U2R: “number of file creations” and “number of shell prompts invoked”
  4. R2L: Duration of connection, service requested, number of failed login attempts

Vals we picked:

**Probing:**

**Low Source Bytes:** Probing attacks often involve sending minimal data to determine the responsiveness or vulnerability of a target system. If the source bytes transferred in a connection are minimal or consist of standardized packets (e.g., SYN packets in TCP/IP for port scanning), it suggests that the connection is more likely part of a probing or scanning activity.

**Repeated Connections:** A series of short-duration connections from the same source IP address or range to multiple ports can indicate systematic scanning or probing to identify vulnerable services.

**U2R:**

**Number of File Creations**:

an attacker typically aims to escalate their privileges to gain root or administrative access on a system. During this process, they might create new files or modify existing ones to achieve their objectives.

Monitoring the number of file creations can help detect suspicious activity. For example, an unusually high number of new files being created by a user account, especially in system directories or areas not typically accessed by regular users, could indicate malicious

Attackers may create new files to install backdoors, upload malicious scripts or binaries, or modify system configuration files to maintain persistence or gain deeper access to the system.

### Tech Stack

What is the Tech Stack:

Frontend: React

Backend: Flask, Python, OpenAI API

Technologies

* Programming Language: Python
* Libraries and Frameworks: Scikit-learn, TensorFlow/Keras, Pandas, NumPy, Matplotlib/Seaborn, React, Flask
* Tools: Wireshark (for network traffic data collection), Jupyter Notebook, Flask (for building a web interface)

We used a Database from online, because we did not have access to any of Nutanix’s data. Each row in the data includes the information abut traffic, including important values such as IP addresses, source bytes, destination bytes,

No ML based, or chatbot based on talking to current Flow Employees

Have you thought about the storage concerns:

* Stored on Nutanix datacenter
* Just Traffic, delete traffic that is a month or more older
* *Because* the most important values are the most current ones, as those are the ones that detect against an attack

Random Forest:

Our attack prediction algorithm leverages a random forest ML model that has been trained on an industry standard dataset.

**Supervised Learning:** training a model on a labeled dataset, where the model learns to map input data (features) to the correct output (labels or predictions)

* Since it’s a networking based product, we would like to incorporate it with Flow. Nutanix Flow is a software-defined networking solution provided by Nutanix. Nutanix Flow enhances the networking capabilities within a Nutanix environment by integrating networking and security features directly into the Nutanix Enterprise Cloud platform.
* Add our dashboard to Prism UI

Improvements:

* Feed our model real time data from Hermes
* From our understanding of the flow networking product, most information regarding packets such as IP and MAC ends up in a Hermes, a proprietary piece of software. Therefore, we believe that this service would be the best feeder into our model.
* Feedback mechanism for false positives
* Make chatbot more specific to Nutanix
* Network admins can directly block an IP
* OpenAI API Privacy issues
* Login with your company ID and only have access to certain IPs

Interface Demo Script:

We start with a couple of widgets on the dashboard. The first one is a pie chart that summarizes the current traffic going in and out, categorizing all of it into either normal, or the specific attack. The second pie chart is similar, and gives an overarching view of the normal traffic versus attacks.

Something special about these charts is that you can filter them by ip address. For example, if there is a destination that has been acting a little weird and seems to be susceptible to an attack, you can enter it’s address, so here we type:

10.15.54.2

and you can see that it been prone to a lot of DOS attacks. Similarly, the most active IPs are shown as well, which can also help with detecting and navigating both traffic and the probaility of attacks on the system.

IF you scroll down, you’ll see our IP connections network graph, which, as explained before, lists all the routes between IP addresses and the protocols they use as well. The graph refreshes every time the page is reloaded to provide up-to-date information on the latest traffic.

Lastly, we have our prediction model.

If you input some important key-value pairs from one of the wireshark entries you have, so in this example, we will

Input:

http as the service used

1032 source bytes,

2 files created,

0 shells, 1 failed login, and a duration of 1,

it’ll send this data in a payload to the random forst model, which will categorize this data, either as normal, or as an attack. Here if we click submit we can see it’s a DOS attack, and because of that result, the values are automatically sent to our chatbot on the right side, asking for advice on how to resolve this issue. You can ask other questions to the chatbot as well, which has been programmed to offer customer support.

The final feature we have is a notification alert system that continuously monitors incoming traffic, and checks if it shows signs of an attack. If it does, the system has a popup, to notify you of the attack so that you can make the necessary precautions.

Project: Anomaly Detection in Network Traffic

Project Overview

The goal of this project is to develop an AI-driven solution that can detect anomalies in network traffic. This can help identify potential security threats or system malfunctions, such as DDoS attacks, unauthorized access, and unusual network behavior.

Objectives

* To collect and preprocess network traffic data.
* To develop a machine learning model capable of identifying anomalies in the network traffic.
* To visualize the detected anomalies for easier analysis and interpretation.
* To implement a real-time monitoring system that can alert network administrators about potential threats.

Technologies

* Programming Language: Python
* Libraries and Frameworks: Scikit-learn, TensorFlow/Keras, Pandas, NumPy, Matplotlib/Seaborn
* Tools: Wireshark (for network traffic data collection), Jupyter Notebook, Flask/Django (for building a web interface)

Steps to Implement

1. **Data Collection:**

* Collect network traffic data using tools like Wireshark, which can capture live network packets.
* Use publicly available datasets like the CICIDS2017 dataset from the Canadian Institute for Cybersecurity.

1. **Data Preprocessing:**

* Clean the data by handling missing values and removing irrelevant information.
* Extract features from the raw network traffic data, such as packet size, source/destination IP addresses, port numbers, and protocols.
* Normalize the data to ensure that it is in a suitable format for machine learning models.

1. **Exploratory Data Analysis (EDA):**

* Perform EDA to understand the distribution and characteristics of the data.
* Use visualization tools like Matplotlib and Seaborn to plot graphs and identify patterns or outliers.

1. **Model Selection and Training:**

* Split the data into training and testing sets.
* Choose an appropriate machine learning algorithm for anomaly detection. Common algorithms include:
* *Isolation Forest:* A popular algorithm for detecting anomalies in a dataset.
* *Autoencoders:* A type of neural network used for unsupervised learning, particularly for anomaly detection.
* *K-Means Clustering:* Can be used to identify clusters and detect outliers that do not fit into any cluster.
* Train the selected model on the training dataset.

1. **Model Evaluation:**

* Evaluate the model’s performance using metrics such as precision, recall, F1-score, and ROC-AUC.
* Fine-tune the model parameters to improve its accuracy and reliability.

1. **Anomaly Detection:**

* Apply the trained model to the test dataset to detect anomalies.
* Visualize the detected anomalies using scatter plots, time series plots, and other appropriate visualization techniques.

1. **Real-Time Monitoring and Alert System:**

* Implement a real-time monitoring system using Flask/Django to continuously monitor network traffic.
* Set up an alert system that notifies network administrators via email or SMS when an anomaly is detected.
* **Deployment:**
* Deploy the model and monitoring system on a server.
* Ensure the system is scalable and can handle large volumes of network traffic.

**Challenges and Considerations**

* **Data Quality:** Ensuring the collected data is clean and representative of real-world scenarios.
* **Model Accuracy:** Balancing false positives and false negatives to minimize both.
* **Scalability:** Ensuring the solution can handle high volumes of network traffic without significant performance degradation.
* **Security:** Protecting the monitoring system from being targeted by malicious actors.

**Future Enhancements**

* **Integration with SIEM:** Integrate the anomaly detection system with Security Information and Event Management (SIEM) tools for more comprehensive security monitoring.
* **Advanced Algorithms:** Experiment with more advanced machine learning and deep learning algorithms to improve detection accuracy.
* **User Interface:** Develop a more intuitive user interface for network administrators to interact with the system and analyze anomalies.

By following these steps, freshers can develop a functional anomaly detection system for network traffic that can significantly enhance the security posture of an organization.

Attacks fall into four main categories:

* DOS: denial-of-service, e.g. syn flood;
* R2L: unauthorized access from a remote machine, e.g. guessing password;
* U2R: unauthorized access to local superuser (root) privileges, e.g., various ``buffer overflow'' attacks;
* probing: surveillance and other probing, e.g., port scanning.

1. Split data into training & testing.
2. truncate training data columns into necessary ones:

Relevant features:

* 1. DOS: “source bytes” and “percentage of packets with errors”
  2. Probing: “duration of connection” and “source bytes”
  3. U2R: “number of file creations” and “number of shell prompts invoked”
  4. R2L: Duration of connection, service requested, number of failed login attempts

1. Train the model
2. Test the model
3. Measure performance, accuracy, ROC, etc for hackathon