Names:

Hannah Emad 2205123

Mariam Mostafa 2205084

Nada Mohamed 2205173

Introduction

We will focuses on the use of big network traffic data to achieve cyber situational awareness and develop a detection model for network traffic anomalies using state-of-the-art algorithms.

Cybersecurity is a major concern for companies and organizations relying on technology, as vulnerabilities can result in millions or billions of dollars in losses. With the exponential growth of Big Data, new techniques and technologies are needed to detect unusual and suspicious activities within the network, allowing for prompt detection and mitigation of anomalous behavior. By leveraging current tools and off-the-shelf state-of-the-art algorithms, organizations can mitigate potential risks and protect their assets.

we are interested in applying Big Data Analytics on enormous network traffic dataset with the goals of generating insights and knowledge to infer and detect unusual and suspicious network behavior

Anomaly Detection: is the process of detecting rare or unusual patterns that deviates from the normal behavior or norm. Unusual patterns are also known as "Outliers" or "Anomalies"

Detection is only possible with existing rules and signatures

- **Establishing a baseline of common patterns Prior to any anomaly detection techniques described above, one of the statistical techniques for anomaly detection would be relying on the commonalities found in the data. One being that the data should follow a normal distribution. Since anomalies don't occur on a regular basis, the assumption is that data should be normally distributed, as majority of the DNS traffic should be largely normal as compared to anomalies. By identifying the baseline of common behaviors/patterns, one can determine what are the common DNS traffic patterns flowing in the network .**
- **The average number of packets and bytes in a normal DNS flow. The term Flows, DNS traffic and DNS flows are used interchangeably in the subsequent sections.**

In addition, if anomalies are presented in the data, it needs to be removed, otherwise the detection model would have failed to detect future instances of anomalies as the detection would assumed it is normal during the detection.

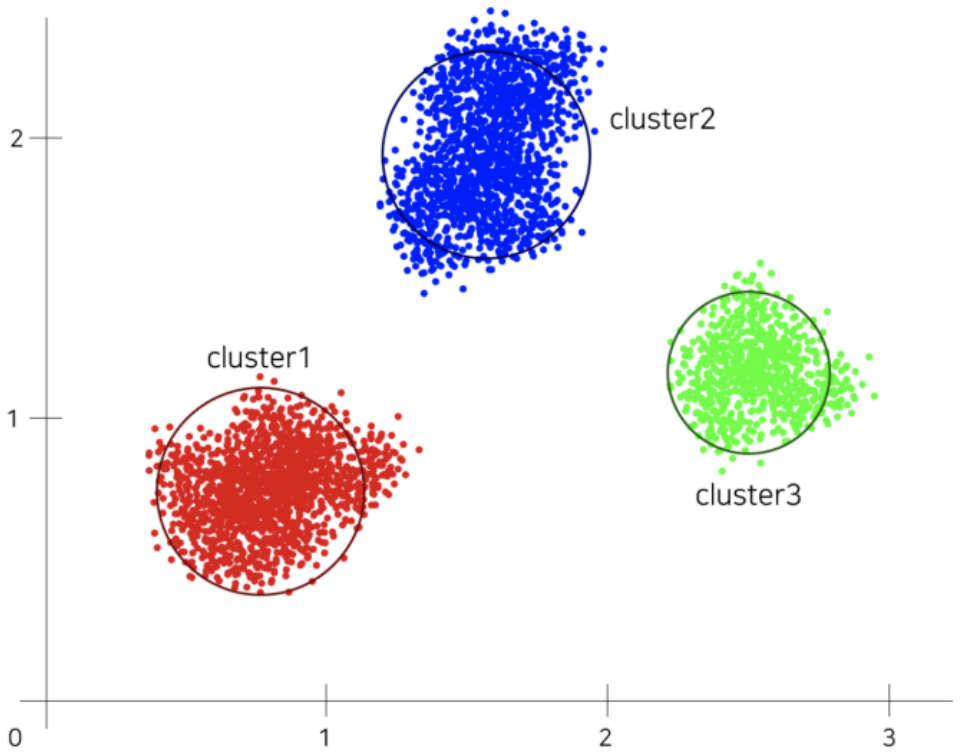
Once the baseline of the common patterns has been established, detection can be deployed using the normal distribution as the baseline. Since anomalies are statistically

different from normal DNS traffic, by modelling the normal distribution of the data, anomalies can be detected one, two or three standard deviations away from the mean.

Clustering in Anomaly Detection

- Clustering is a common technique used to group similar objects together for cluster analysis. Objects that are similar to each other belong to a cluster of it's own and vice versa for dissimilar objects.
- For anomaly detection in DNS traffic, similar DNS traffic patterns should belong to a cluster of its own, using some distance metrics such as the Euclidean distance. By following this rule, anomalies can be detected when new and incoming patterns stray away from any of the clusters or its distance to any of the clusters exceeds a certain threshold. It has been proven that clustering algorithms shows promises by learning distinctive and complex patterns from data without human intervention.

Display three points of groups



Tools and framework used for the proposed anomaly detection workflow

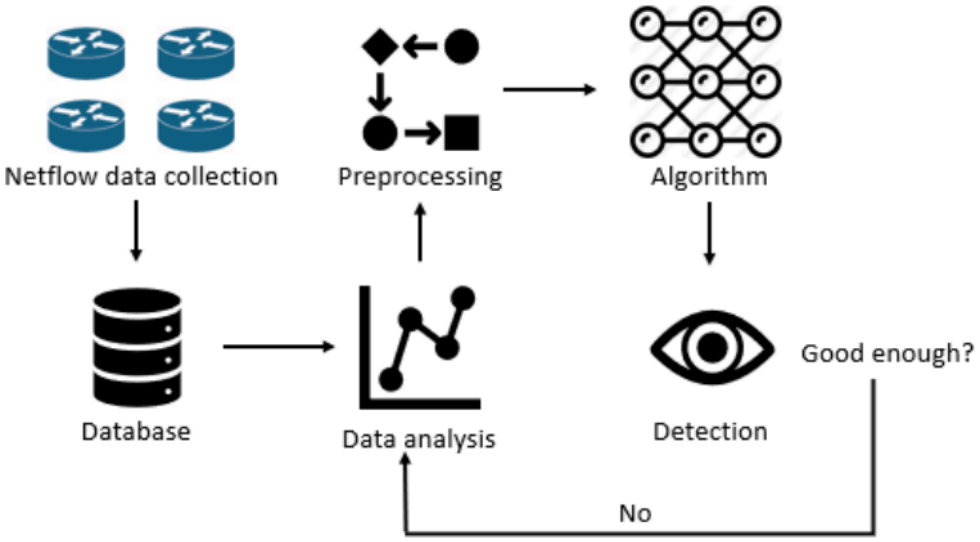
Tools and Framework : Large scale data processing using Big Data Analytics are performed using the following big data cluster. The hardware specifications for the big data cluster consists of 9 nodes with a total of 544 virtual cores and 3.5TiB of memory.

Proposed Anomaly Detection Workflow : The proposed anomaly detection workflow first comprises of

- **Data collection**

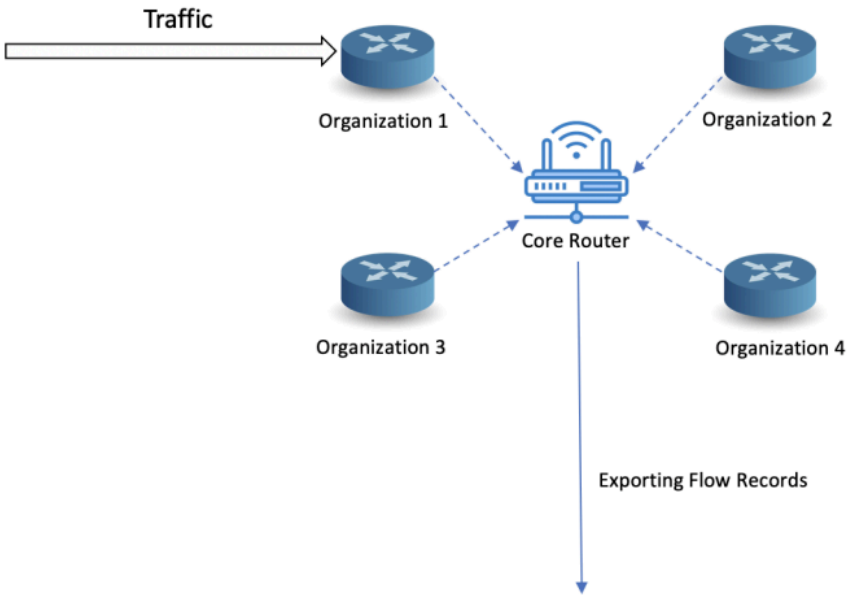
- Data preprocessing
- Training using clustering algorithms
- Deploying model for detection
- Evaluation
- Reiterate

Anomaly Detection Workflow



Exploring DNS Traffic through NetFlow Data Analysis

NetFlow : Is a traffic monitoring protocol developed by Cisco for collecting network traffic flows from NetFlow-enabled router. Data collected using NetFlow can be used by network analyst to understand how network traffic is flowing in and out of the network.



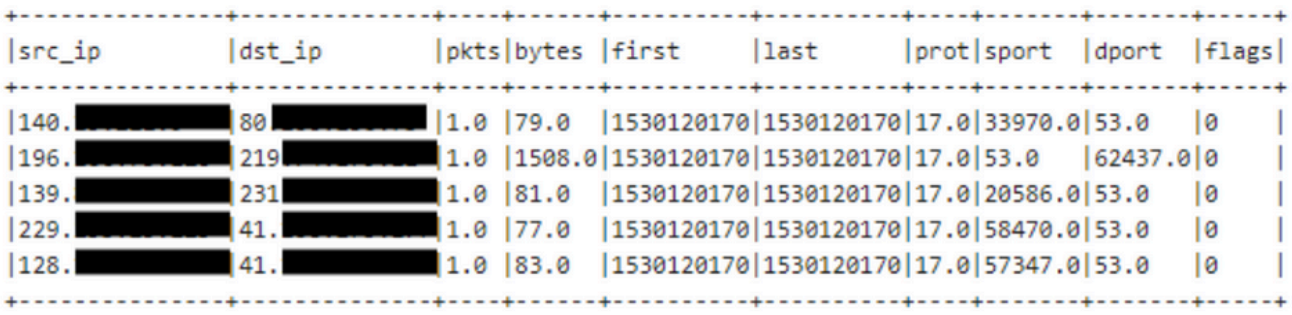
NetFlow data are collected from various multi-enterprise network

Example of : Working with NetFlow data to focus on DNS traffic for anomaly detection.

The NetFlow dataset used in this research was provided by an industrial partner. Originally, the dataset had 48 attributes and contained one day's worth of network traffic with 253 million records. For this study, it was reduced to 10 key attributes and filtered down to 4 million records, focusing only on DNS traffic. Other services like HTTP and SSH were excluded.

In the dataset, called a Network Flow, represents a transaction between a source and destination address. The goal is to use this data to identify DNS anomalies.

F = (IP_src,IP_dst, pkts, Bytes, T_start, T_end, rot, Port_src , Port_dst, Flags)



src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

DNS flow packets count

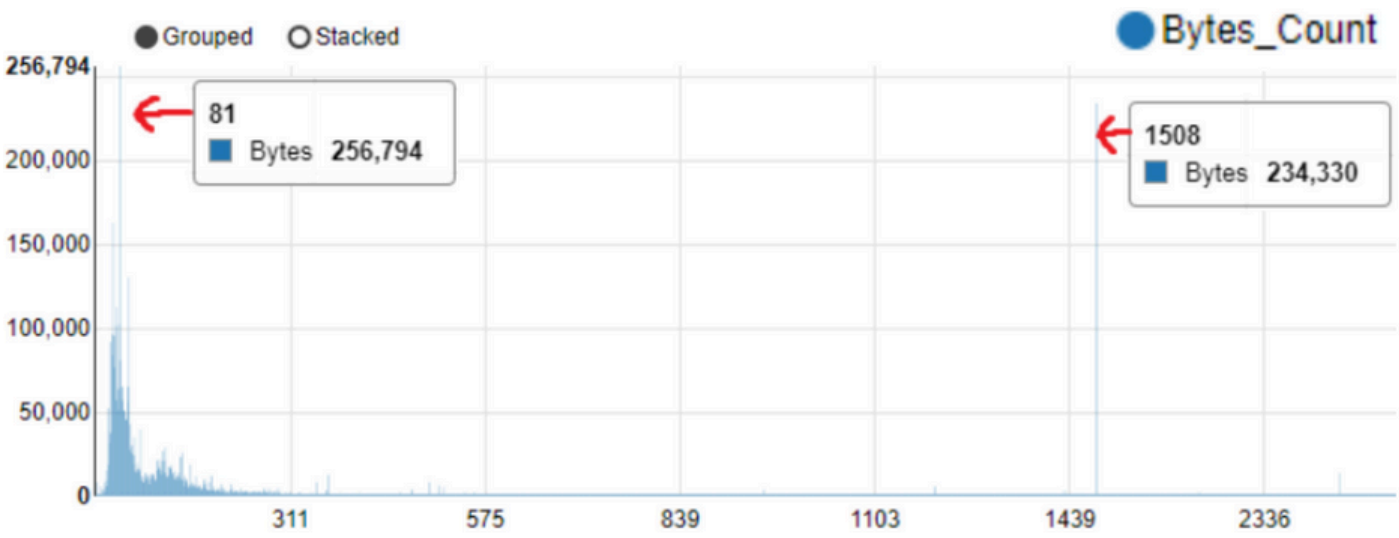
shows packet feature where packet 1 and packet 2 are the most common number of packets send per DNS flow as compared to the next subsequent leading packets 3..5, hence 4.1 million counts of packet 1 formed a commonalities of 95% of the total DNS flow.

packet	UDP	TCP
1	4147021	274610
2	82152	1177
3	22477	12
4	11147	95
5	6753	70

More than 95% of the total DNS flow are formed by only packet 1 & packet 2

Normality of Data: The above shows the 5 most relevant NetFlow attributes, the most useful features or attributes for determine the normality and distribution of the data are the no. of pkts per DNS flow, the pkts feature, as shown in TABLE I above, many of the UDP flow and TCP flow largely contains only 1 packet per DNS flow. Observations of UDP DNS flow with low packets count are much more common than larger ones.

Shows the typical distribution of the bytes feature commonalities in the NetFlow:



Feature Selection: Selecting the relevant features is an indispensable process before data preprocessing, since the goal is to retain as much information as possible and remove any redundant information from the dataset that does not constitute towards the detection of anomalies. The feature pkts is one important feature, which helps in the detection of anomalies. It is evident from our initial analysis, DNS traffic with packets count between 1 and 5 made up of more than 95% of the data commonalities.

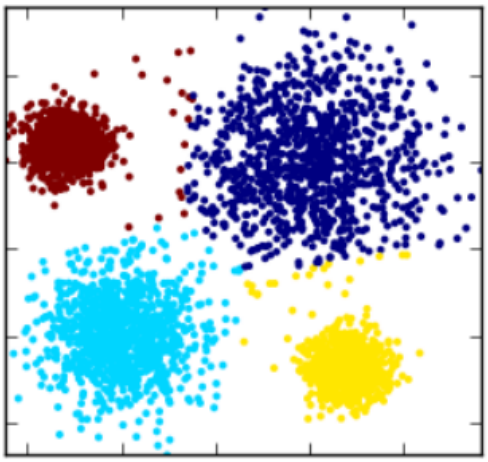
Finalize Feature Selection : These 10 features we originally had in the NetFlow dataset. These features will be used for data preprocessing and implementing the anomaly detection model.

src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

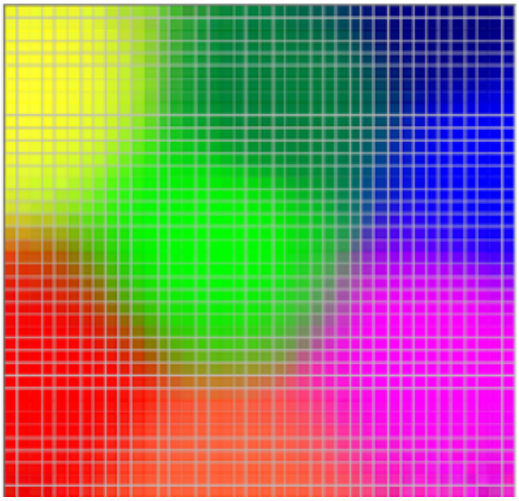
Types of Clustering Algorithm : There could be a better clustering algorithm more suited for

- **K-means clustering - Based on centroid models.**
- **Self Organizing Map (SOM) - Based on unsupervised neural network model with competitive learning.**
- **Gaussian Mixture Model (GMM) - Based on distribution and probabilistic models.**

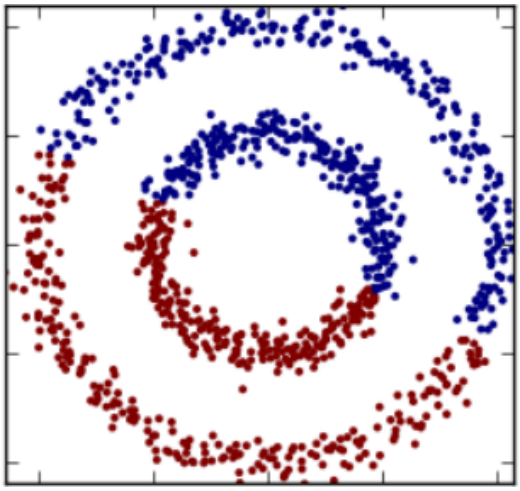
K-means:



SOM:



GMM:



Implementing the anomaly detection model

we will discusses the preprocessing stage of the anomaly detection model, and finally the inherent limitation and caveat of the aforementioned three clustering algorithms. Both UDP and TCP DNS flow will be trained on clustering algorithms after data preprocessing.

Initial features: some features underlined in red, and were used to produce additional features through aggregation these categorical values the ML algorithms.

Features selected for data preprocessing:

src_ip	dst_ip	pkts	bytes	first	last	prot	sport	dport	flags
140.[redacted]	80.[redacted]	1.0	79.0	1530120170	1530120170	17.0	33970.0	53.0	0
196.[redacted]	219.[redacted]	1.0	1508.0	1530120170	1530120170	17.0	53.0	62437.0	0
139.[redacted]	231.[redacted]	1.0	81.0	1530120170	1530120170	17.0	20586.0	53.0	0
229.[redacted]	41.[redacted]	1.0	77.0	1530120170	1530120170	17.0	58470.0	53.0	0
128.[redacted]	41.[redacted]	1.0	83.0	1530120170	1530120170	17.0	57347.0	53.0	0

Preprocessed features: The dataset underwent preprocessing to aggregate network flows intervals based on features like source IP and timestamps. This approach ensures flows adhere to DNS service norms, such as not exceeding a specific packet count.

For example:

The source IP "231.x.x.x" communicated with 133 unique destination IPs, exchanged 297 packets (33.6k bytes), used one source port to reach 294 destination ports via protocol 17 (UDP), and generated 295 flows within one minute. This IP belongs to a legitimate DNS server (e.g., Google) and exhibited normal behavior by resolving DNS requests to multiple destination IPs using source port 53.

src_ip	unique_dst_ip	no_pkts	no_bytes	no_sport	no_dport	prot	no_flow
231.[redacted]	133	297.0	33654.0	1	294	17.0	295
213.[redacted]	5	23.0	2938.0	20	5	17.0	23
41.[redacted]	606	1387.0	545268.0	1	1193	17.0	1326
13.[redacted]	808	1774.0	127728.0	1604	1	17.0	1630
13.[redacted]	583	864.0	62222.0	815	2	17.0	817

Feature Scaling: Feature scaling consists of normalization transform data into a specific range, into the range ”[0,1]” .

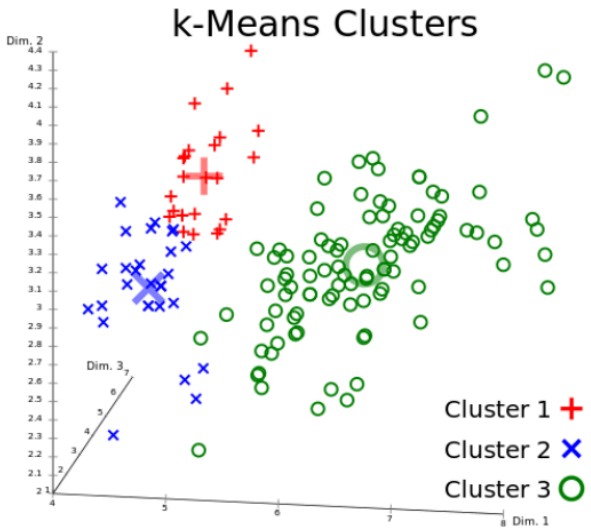
Dimensionality Reduction: After feature scaling, dimensionality reduction techniques like PCA are applied to simplify data visualization and analysis, reducing complexity to two or three dimensions for better interpretability.

The preprocessed features were transformed into six principal components using PCA.

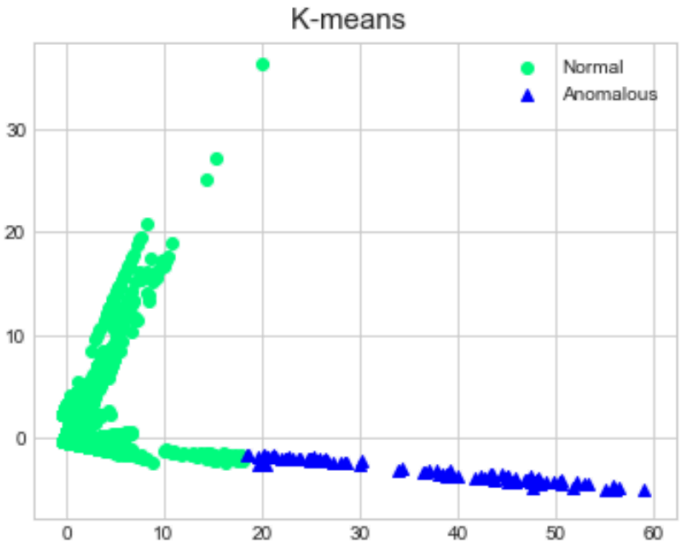
Clustering Algorithms

K-means Clustering: K-means is a widely-used unsupervised clustering algorithm known for its efficiency in handling large datasets with O(n) complexity. It is highly scalable and often

used for preliminary cluster analysis. The algorithm works iteratively and requires the user to specify the number of clusters (k), also known as centroids.



The K-means algorithm is trained with the following parameters: k = 2 , a maximum of 100 iterations. With no labeled data, the assumption is made that the dataset can be divided into two clusters: Cluster 0 representing normal traffic and Cluster 1 representing anomalous traffic. In the analysis of DNS flow.



Anomalies Detection after Model Selection:

After selecting the optimal number of components based on the lowest BIC and AIC scores, the Gaussian Mixture Model (GMM) was re-trained with five components and a "Full" covariance type.

Model estimation techniques using BIC/AIC has been selected to determine the optimal number of components/clusters for the GMM model. Using only five components, the final GMM model achieved a 100% detection

The GMM successfully detected anomalies in the simulated NetFlow dataset, where Cluster 1 represented normal traffic (81% of DNS flow) and Clusters 2 to 5 indicated anomalous or

unknown traffic.

DNS anomaly detection with a focus on cache poisoning

Focus on Cache Poisoning Detection

Cache poisoning is a critical DNS vulnerability where attackers inject malicious data into DNS resolvers' cache, redirecting users to malicious websites. To adapt the anomaly detection workflow: - Key Indicators of Cache Poisoning: - Mismatched or unexpected IP responses for known domains. - Abnormal changes in TTL (Time-To-Live) values. - High entropy in transaction IDs or non-standard query patterns.

Key Indicators of Cache Poisoning:

- Mismatched or unexpected IP responses for known domains.
- Abnormal changes in TTL (Time-To-Live) values.
- High entropy in transaction IDs or non-standard query patterns.

Implementation Strategy:

Incorporate anomaly detection tailored to cache poisoning

- Monitor TTL values over time. Legitimate DNS entries typically have stable TTLs.
- Compare transaction IDs to identify patterns inconsistent with standard randomness (using Shannon entropy).

Transaction ID Entropy Analysis

Transaction ID (TXID) entropy is a crucial metric for identifying cache poisoning attempts, as attackers often guess transaction IDs to inject malicious data.

Steps for Implementation:

Calculate Shannon entropy for the transaction ID field across DNS queries.

Use thresholds to flag anomalies where entropy values deviate from the expected randomness.

Synthetic Data Generation

In the absence of real-world DNS traffic data, generating synthetic data is essential for training and evaluating the detection model.

Steps:

Use tools like dnsperf or mgen to simulate DNS traffic.

Introduce controlled anomalies such as

- Responses with mismatched query and answer sections.
- Repeated queries with manipulated transaction IDs.

Deployment of the Detection

Model Effective deployment ensures that the detection model transitions from the experimental phase to real-world application. Deployment

Steps:

Stream Processing

- Use tools like Apache Kafka or Spark Streaming to process DNS logs in real time
- Integrate with SIEM (Security Information and Event Management) systems for alerting.

Model Integration

- Deploy trained models using REST APIs.
- Continuously update models with new data to adapt to evolving threats.

Revised Workflow

Data Collection: Real-time DNS traffic or synthetic datasets with labeled anomalies.

Feature Extraction: TTL, query volume, transaction ID entropy, source/destination IP patterns.

Model Training: Use clustering algorithms (K-Means, GMM) or supervised models for anomaly detection.

Evaluation: Test on simulated cache poisoning scenarios.

Deployment: Real-time detection and integration with network monitoring systems

---

Conclusion

we have explored the application of anomaly detection techniques to DNS traffic, focusing on identifying unusual patterns that may indicate potential security threats such as cache poisoning. The combination of Big Data analytics and state-of-the-art machine learning algorithms, particularly clustering algorithms like K-means and Gaussian Mixture Models (GMM), shows promise in detecting abnormal network behavior in large-scale datasets.

The methodology begins with data collection from NetFlow, followed by preprocessing, feature selection, and transformation. The application of clustering algorithms for anomaly detection enables the identification of patterns that deviate from the normal behavior of DNS traffic. This approach allows for effective detection of network intrusions, with a particular focus on malicious activities such as DNS cache poisoning.

The proposed anomaly detection workflow involves several steps, including data collection, preprocessing, training the model using clustering techniques, and evaluating the model's performance. A key aspect of the detection is the use of normality assumptions based on DNS traffic patterns, where anomalies are defined as deviations from established norms. By leveraging clustering techniques such as K-means and GMM, we can effectively categorize DNS traffic into normal and anomalous clusters, enabling the identification of suspicious traffic that may require further investigation.

The integration of anomaly detection with real-time DNS traffic monitoring is a crucial step toward securing network infrastructure. Tools like Apache Kafka or Spark Streaming allow for real-time analysis, and the deployment of models via REST APIs ensures continuous monitoring and updating of the detection system. Additionally, the incorporation of key indicators for cache poisoning detection, such as abnormal TTL values and transaction ID entropy, enhances the model's effectiveness in identifying malicious behavior.

The ability to generate synthetic data for training purposes further strengthens the model's robustness, as it allows the simulation of potential attacks under controlled conditions. This is particularly valuable in environments where real-world DNS traffic may not be readily available.

Overall, the proposed anomaly detection system not only offers a comprehensive solution for DNS traffic analysis but also lays the groundwork for broader applications in network security. By continuously refining the detection model and adapting it to new threats, organizations can better protect their networks from emerging cyber threats, ensuring the security and integrity of their online services.