# Introduction

Digital transformation, which is taking place at an accelerated speed at companies and organizations around the world, puts an increasing emphasis on the need to protect digitized information and components of digital technology against the rising threat of cyber security attacks. Most of the time, the ultimate objectives of cyber criminals are financial gains, espionage, or disruption of operations. No company, organization, or even state is safe from their penetrating power.

Cyber security professionals are engaged in the asymmetric warfare, constantly on defense against cyber criminals, who continually devise new ways to penetrate the cyber security systems. Just as it is impossible to anticipate all inventions of the human, albeit criminal, mind, it is impossible to stop all cyber attacks. That doesn’t mean we stop trying, rather that that we build another line of defense to deal with successful breaches to minimize the damage.

According to IBM Cost of Data Breach Report 2020 “The average time to identify and contain a data breach, or the ‘breach lifecycle’, was 280 days in 2020. Speed of containment can significantly impact breach costs, which can linger for years after the incident.” The reasons for such a slow response vary and include such factors as

* Ever increasing sophistication of the APT (advanced persistent threat) techniques
* Ever increasing number of endpoints through which attackers can reach their targets
* Lack of efficient security log management strategy

While we cannot stop 100%, or anywhere near 100%, of all attacks, we should strive to detect all the breaches and to minimize the detection time.

# Description

Currently, organizations use multiple cyber security solutions for attack detection. Each solution produces one or several signals that may be indicative of a cyber attack.

These signals, henceforth referred to as primary signals, come from Next Generation Firewalls (NGFW), Intrusion Detection Systems (IDS), Endpoint Detection and Response (EDR) platforms, etc.

The primary signals have a high false positive rate. A large volume of primary signals overwhelms SOC analysts. By some accounts the average enterprise now has more than 75 different security threat detection tools in their overall cybersecurity stack. This creates a possibility of real signals getting unnoticed, lost in the sea of false alarms.

To deal with this challenge, the primary signals must be correlated and combined into secondary signals. The correlated secondary signals have a lower probability of being false positives.

Currently, the process of correlation is done using known patterns/signatures. The patterns are created using historical knowledge of previous attacks. As a result, this approach cannot detect any new attacks. Because it is manual, the process is time consuming and prone to errors.

We propose a system that uses graph technology to perform signatureless correlation of the primary signals in order to produce high confidence, high probability actionable alerts.

Graph is a collection of nodes and edges, where edges represent relationships between the nodes. One can use various graph analytics metrics to characterize graphs. Some examples of graph metrics are betweenness centrality, closeness centrality, eigenvector centrality, edge connectivity, node connectivity, graph distance. Graph can be characterized by the number, sizes, connectivity of communities within each graph. Another type of numeric representations of graph is created using graph embedding. A non-inclusive list of possible graph embedding techniques are node2vec [Ref1], DeepWalk [Ref2], Deep Neural Networks for Learning Graph Representations (DNGR) [Ref3].

Cyber security data lends itself naturally to be represented by graphs, be it packets sent between different IPs, emails sent between users, or login attempts by a user.  The tool will convert multiple primary signals for predefined time periods, e.g., on hourly basis, into graphs. As a result, all the network suspicious activities will be converted to a sequence of graphs. The standard machine learning algorithms for outlier detections, such IsolationForest, Autoencoder, DeepSVDD, will be applied to various numeric representations of the graphs. The outliers found by these algorithms will constitute the secondary signals. The secondary signals will have a higher probability to be true positives than the primary signals that were used to create them.

Our system performs the following steps (Figure 1):

1. Ingest, normalize, and aggregate various security logs

2. Extract weak primary signals from individual channels

3. Combine weak primary signals to create secondary signals with much higher probability of being malicious attacks, as well as calculating the severity of the attacks

## Ingest and Extract

The raw logs are collected from various sources, such as EDR, IDS, NGFW, etc., are stored in a NoSQL database, such as document database MongoDB.

The raw logs are parsed, and the parsed results are stored in a SQL database, such as MySQL.

Primary signals, or events, created upstream by various tools (NGFW, IDS, EDR, etc.) are extracted from the logs. Some examples of such events are a potentially malicious PowerShell command executed on a laptop of an employee, an attempt to connect to a malicious website from the laptop, a number of failed authentication attempts for the user within a short period of time

The primary signals are used to create graphs using such tools as Neo4j graph database or networkx python package. The graph data is stored in a graph database. (Figure 3)

The graph data is further processed to create features for ML and DL. Depending on the environment the features are stored either in a SQL database or in parquet files.

## Combine

The main ingredient of our system is the process of correlation of various primary signals and combining them into high confidence, high probability actionable alerts.

It is done using graph analytics and anomaly detection.

The system consists of two different graphs: short term and long term.

A short term graph contains all the primary signals collected over a predefined fairly short time window, e.g. 1 hour or 2 hours. Once a short term graph is processed (as described below) it is discarded and new short term graph is created for the next time window. The time windows may or may not overlap. An example of non-overlapping time windows is 1PM – 2PM, 2PM – 3PM, 3PM-4PM, etc. An example of overlapping time windows is 1PM – 2PM, 1:30 PM – 2:30PM, 2PM-3PM, 2:30PM – 3:30PM etc.

The long term graph persists through time. It gets updated using some of the elements - nodes and edges - from the short term graphs. Over time the components of the long term graph will be different, but the graph itself is never explicitly destroyed.

Each entity is a node in the graph, different events create relationships (links) between nodes. In the above example the employee, the laptop, the website, the authentication servers are node, the events connect those nodes using the appropriate relationships.

The long-term graph contains clusters built over time from the potentially malicious clusters in the short-term graphs. Each cluster in the long-term graph is assigned a probability of being malicious.

The following describes the process of adding new clusters from the short term graphs to the long term graph:

For each new short-term graph

1. Perform clustering (see section 1.4). Multiple clusters are created as nodes connected by relationships.
2. Perform anomaly detection. The system keeps track of different clusters using various graph analytical metrics. It creates a profile of ‘normal’ clusters and compute and quantify the deviation of any new clusters from the norm. Anomaly detection process is depicted in Figure 5.
3. A configuration with the profile significantly different from a set of normal profiles is assigned a calculated probability of being malicious. The probability value reflects the extent of deviation of the configuration profile from a set of normal profiles. If there are anomalous clusters compute their probability of being potentially malicious.
4. Clusters with the profile significantly different from a set of normal profiles (potentially malicious cluster) are added to the long-term graph (Figure 6a. and 6b.)
5. When a new cluster from a short term graph is added to the long term graph, the following logic is used to update the probability of clusters being malicious.

a. If there is any overlap of the newly discovered potentially malicious cluster with the existing/current clusters in the long-term graph, combine them and update the probability of the combined cluster of being malicious (equation 1 is one possible formula for updating probabilities, other options can be used)

b. Otherwise just add this cluster with the associated probability to the long-term graph

Equation 1

Here is the probability of the current cluster to be malicious, is the number of nodes in the current cluster, is the probability of the new cluster to be malicious, is the number of nodes in the new cluster, is the number of nodes common to both current and new clusters.

This approach allows tracing prolonged malicious attacks that may exhibit periods of no activity.

Whenever the probability of being malicious exceeds the optimized threshold, a response is triggered. The response will lead to stopping of the detected malicious activity. The cluster corresponding to this malicious activity will be removed, i.e. the nodes and edges of the cluster will be removed from the long term graph.

Figure 7 Illustrates the evolution of the long term graph. Starting with the first time frame which contains one cluster, new nodes and edges are added to the first and then to the second cluster. As the nodes are added the probability to be anomalous Pr\_anom is growing. When Pr\_anom exceeds the threshold, which in this hypothetical case is equal to 0.9, a response is triggered, malicious activity is interrupted, and consequently, Cluster 1 is removed from the long term graph.

## Anomaly Detection

A number of metrics is calculated for each short term graph, including but not limited to Betweenness Centrality, Closeness Centrality, Eigenvector Centrality, Edge Connectivity, Node Connectivity, Number of Communities, Communities Size Distribution, Node2vec node embedding, DeepWalk node embedding, DNGR node embedding. The metrics are used as input for several anomaly detection algorithms. The results of the anomaly detection algorithms are combined using some variation of the Majority Voting algorithm.

## Clustering

Clustering is a process of separating a graph into a number of subgraphs, where the nodes within each subgraph are closer to each other according to some measure than to the nodes in other subgraphs. There are a number of algorithms that can be used for graph clustering depending on the nature of the graphs. These include but limited to Louvain, Label Propagation, and K-Means clustering.

# References

Ref1 <https://arxiv.org/pdf/1607.00653.pdf>

Ref2 <https://arxiv.org/pdf/1403.6652.pdf>

Ref3 <https://paperswithcode.com/paper/deep-neural-networks-for-learning-graph>

# GLOSSARY

EDR - Endpoint Detection and Response

NGFW - Next Generation Firewall

DNGR - Deep Neural Networks for Learning Graph Representations

DeepSVDD - Deep Support Vector Data Description

ML - Machine Learning

DL - Deep Learning

DNN - Deep Neural Networks

GNN - Graph Neural Networks

# List of Figures

Figure 1. The Processing Flow

Figure 2. Data Transformation Chain

Figure 3. From Primary Signals in Log data to the Short-Term Graph

Figure 4. Schematic Graph Processing

Figure 5. Anomaly Detection

Figure 6a. Before Merging

Figure 6a. Before Merging

Figure 7. Evolution of the Long Term Graph