Early Detection of Lateral Propagation of Ransomware in Industrial/OT Environments

1st Satish Mohan
CTO
Airgap Networks
Santa Clara, United States
satish@airgap.io

2nd Emre Yavuz
Intern
Airgap Networks
Santa Clara, United States
eyavuz@umich.edu

3rd James DeBoskey
Intern
Airgap Networks
Santa Clara, United States
jddebosk@berkeley.edu

***Abstract*—This document is a full description of Airgap Networks's Ransomware Early Detection (RED™) Algorithm, using a Hierarchical Density-Based Spatial Clustering algorithm [1] to identify potentially malicious outliers in data logs.**

***Index Terms*—Ransomware, Clustering, Data, Malware, Security, Algorithm**

I. Introduction

Ransomware is one of the biggest threats facing the security industry today. Businesses have deployed multiple layers of security solutions to defend themselves against ransomware attacks. However, despite this, these attacks continue to occur, and enterprises find themselves in a situation where they either must pay the demanded ransom or risk losing access to critical business assets and data.

Early detection is of utmost importance regarding ransomware, as it can cause potentially irreversible damage to mission critical business assets. This document summarizes the algorithmic enhancements we made to the popular HDBSCAN (Hierarchical Density-Based Spatial Clustering of Applications with Noise) clustering model used to detect anomalous and malicious behavior in data logs, so that we can detect network lateral movement attempts typically adopted by ransomware. The scope of these algorithmic enhancements are restricted to Industrial/OT environments where network traffic behavior is very predictable and anomalies are easy to identify. Further we restrict the analysis to east-west (lateral) traffic between endpoints within the Industrial/OT environment and exclude any north-south traffic (to public internet or external locations).

Modern ransomware attack techniques involve lateral movement on the network, which refers to the attacker's movements once they've compromised an endpoint within the network and gained initial access. With Lateral movement, attackers can move from system to system in the network, using various attack tools to gain elevated trust and harvest credentials until they are able to compromise critical business assets. With ransomware lateral propagation on the rise, it is of paramount importance that these threats are identified early in order to contain their blast radius and prevent any private information from getting into the hands of the attackers.

Airgap's Ransomware Early Detection or RED is an extension to the Airgap Zero Trust Isolation Platform. RED uses a combination of AI-based clustering models and deep packet inspection based behavioral analysis models, and integrates with Airgap Ransomware Kill Switch™to automatically quarantine compromised devices and disable access to mission critical business assets

II. The RED AI/ML Analysis Pipeline

The current RED AI/ML pipeline is a 7-stage Software as a Service (SaaS) pipeline utilizing the AWS "serverless" ecosystem's micro-services for both computation and intermediary data storage.

A. The Primary Level

Network flow data from Industrial/OT customer network environments is collected, summarized and sent into AWS's Kinesis Firehose micro-service. AWS Kinesis Firehose bulks the incoming flow data for a set of time intervals and pushes a bulk payload at the end of said interval into the AWS Simple Storage Service or S3. The raw network flow data is archived here for later customer retrieval and inspection, but is also fed into the AWS Batch computing service for Primary-Level Clustering Analysis (thereby forming a sub-pipeline of 3 stages). Here network traffic chunks are grouped by origin and destination IP pairs to compute network clustering anomalies on a per-edge basis (Analogizing a customer's network to a dense graph of both internally and connected externally to the internet). The features used for clustering along each network edge are total bytes sent/session, total bytes received/session, average bytes sent/session and average bytes received/session. Clusters are computed in the primary level and all anomalous network activity are recorded in AWS DynamoDB as new-connection and volumetric anomalies. This data is then pushed into the secondary-level pipeline (which is another sub-pipeline of 3 stages) for secondary level clustering.

B. The Secondary Level

In the secondary level, a similar clustering is now performed on network anomalies on a per-site rather than per-edge basis, where dense clusters are labeled and pushed onto the final stage of the pipeline, AWS DynamoDB. These dense clusters recorded in DynamoDB are retrieved by the Airgap user interface and presented to the customer on a heat-map as likely

malicious behavior or anomalous patterns of network behavior. Throughout the pipeline, AWS Lambda is used as connective "tissue" to help reformat data for transfer, push intermediary logs picked up by malware static analyses detector tools such as zeek to the front-end, along with aid in micro-service communication and isolation.

C. Goals

Overall the pipeline aims to provide a low-latency and low-overhead tool to provide client-less analyses of network traffic without ever impeding the flow of said traffic, but sufficiently fast to alarm customers of malicious network behavior upon the first set of lateral proportions throughout a user's network.

III. Algorithm Overview

A. Pre-Processing

This application can be used as a way for consumers to detect any abnormal and potentially malicious behavior on their network. The algorithm takes in thousands of data entries and filters them to avoid high levels of redundancy and unnecessary calculations. From there, the algorithm extracts four features: orig bytes, resp bytes, mean orig size, and mean resp size, to have quantifiable measures to cluster the data. The algorithm use these four features specifically, since using extra features could result in our algorithm failing to the curse of dimensionality. These features provide sufficient information to classify each point. After the feature extraction, the data is noramlized on a logarithmic scale, so our clustering classification can identify and classify extremities with a suitable score.

B. Algorithm Procedure

Once the pre-processing is completed, the data goes into the HDBSCAN algorithm, which builds a minimum spanning tree from the inputs, creating a clustering hierarchy based on its connected components.

The algorithm then uses the soft clustering functionality given by the HDBSCAN library and score the anomalies based on their likelihood of being in a cluster if they aren't in a cluster already. From there, these anomalies are run through the algorithm once more, and by using the soft clustering algorithm, scores are given to these anomalies based on their likelihood of not being in a cluster. Using the soft clustering rather than the hard clustering allows us to give fair scores to points on the boundaries of clusters, as the end result becomes a vector of probabilities instead of a hard defined cluster that the point is associated with.

It is also important to note that every new IP pairing in the logs is marked as an anomaly to ensure that traffic is intentional.

C. Why HDBSCAN?

The three main criteria for choosing a clustering algorithm were having an algorithm that could manage varying shapes, manage varying densities, and provide robust outputs. The algorithm had to be able to adapt to the sporadic nature of network traffic. There are many clustering algorithms that could have provided decent results, but our team decided to use a density based clustering algorithm since we believed it was the best way to identify patterns in network traffic. This style of clustering easily identifies noise, and groups together points that are highly dense. This translates really well to finding routines in user behavior, and pinpointing abnormalities. The reason for using a hierarchical DBSCAN over the standard DBSCAN is that although regular DBSCAN does very well when it comes to picking out clusters of various shapes, it falls short when it comes to managing clusters of varying densities. We decided to go with HDBSCAN for its ability to manage clusters with varying densities, and allows us to decide which clusters were important based on size. This is a critical issue, as network traffic doesn't emulate a uniformly distributed density across the charts; there are some areas that will naturally be more dense than others. The nature of network traffic varies from user to user, so we needed an algorithm that could adapt accordingly.

IV. Testing Procedure

We ran the MS17-010 EternalBlue exploit as well as a few of our own encryption and decryption algorithms on several windows 7 VMs (Virtual Machines) to test our algorithm. Our setup to run these programs consisted of one Kali Linux VM with 2 vCPUs, 50 GB of storage, and 8 GB of memory; three Windows 7 VMs using the SMBv1 file sharing protocol to allow the EternalBlue exploit to work with 2 vCPUs, 32 GB of storage, and 2 GB of memory.

To generate normal traffic, we used several scripts to ping each VM, get the VMs to access certain websites, download files, and share files across VMs. These would run every few minutes on an infinite loop.

To generate anomalous traffic, every 8 hours, we would run one of the following: 1) Running the EternalBlue exploit on each of the VMs, 2) Uploading an encryption and decryption algorithm on the Windows 7 VMs from the Kali Linux VM, and running the algorithm from a mounted drive, or 3) Running a scan to detect which VMs were vulnerable. We also ran a script that would share files to the VMs with suspicious filename endings, such as ".covid" or ".2spyware", every 8 hours.

Creating such a mixture of traffic allows us to simulate what an average user does on their computer and test to see if our algorithm can differentiate between what is safe behavior and what is potentially malicious.

V. Results

After running our algorithm over the course of two weeks, we found that the normal traffic wasn't creating any alarms on the UI, but the occasional exploits, encryptions, and port scannings were.

We took advantage of the idea that ransomware is most likely a very small or nonexistent portion of a user's traffic, so the algorithm was able to understand what normal traffic looks like and easily identify any traffic that didn't fit the mold.

Here is the resulting plot in 3 dimensions for ease of viewing.

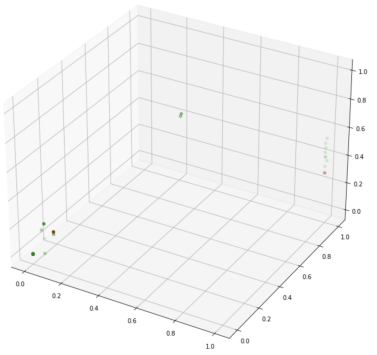

Fig. 1. 3-D Plot of Data Features

The following figure contains the features of two anomaly points that resulted from intermittently running the Eternal-Blue exploit's port scans alongside normal traffic.

	orig_bytes	resp_bytes	mean_orig_size	mean_resp_size
0	0.041678	0.067539	0.076920	0.133454
1	0.041943	0.066666	0.076808	0.129858

Fig. 2. Outliers from the R.E.D. Algorithm

Most NDR (Network Detection and Response) vendors emphasize traffic entering and exiting the network, but lack resources for uncovering lateral movement. Airgap's R.E.D. can be a useful tool for adding another line of defense against ransomware lateral movement. By using an algorithm that learns a user's baseline network traffic, users can detect anything abnormal that occurs immediately after it happens. In addition, R.E.D's integration with Airgap's Ransomware Kill Switch allows users to stop the spread of ransomware before it has the chance to spread through the entire network.

REFERENCES

[1] L. McInnes and J. Healy, "Accelerated hierarchical density based clustering," in *Data Mining Workshops (ICDMW), 2017 IEEE International Conference on*. IEEE, 2017, pp. 33–42.