Intro

* Why are we doing this
* Why we used cuckoo
* How we used cuckoo

Python Scripts

We needed an executable to give to Cuckoo to process and give us the raw data. The first problem we encountered was Cuckoo not accepting our executable. We used pyinstaller [2] to turn our script into a .exe, but Cuckoo gave errors because of the python version, and at the time, we used python 3.11. The highest python version that cuckoo can use is python version 2.7 [3]. After going to python 2.7, we lost packages that were not available for python 2.7.

Our solution was to use python 2.7 and simplify our python script and disregard turning it into a .exe but give it to cuckoo as a .py file.

Data (Cuckoo Logs, PCAP Logs)

After giving cuckoo our script, cuckoo gave us the three files, analysis log, behavioral log, and PCAP log.   
Analysis logs are the logs that cuckoo generates when a file, processes, and errors are created; behavioral logs are analysis logs in more detail; and PCAP logs have the network traffic.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 1.1 Raw Analysis Log

Calendar

Description automatically generated with low confidence

Figure 1.2 Raw PCAP Log

Figure 1.3 Raw Behavioral Log

Given this data, we need to process and clean up for the machine learning algorithm to make sense of the data.

Processed Data

To process our analysis data, we had to filter anything that was not a valid path. Using regex to filter and save a txt file it was then possible to turn the paths as a term for TF-TDF.

Text

Description automatically generated

Figure 2.1 New Analysis Log

We decided that past encryption was unnecessary since we are trying to warn when it detects a virus before encryption. Thus, we removed data when it showed signs of when WannaCry encrypted.

Text

Description automatically generated

Figure 2.2 Pre-Encryption Log

Since WannaCry’s encrypted files have the file extension .WNCRYT, we used regex to find when we encountered .WNCRYT and stop appending it to the new log, Pre-Encryption log.

We noticed that each path is different for every scenario, we decided on two ways to combat this, one where we generalize the paths and to remove the paths completely. Figure 2.3 and Figure 2.4 are generalized and removed logs respectfully.

Text

Description automatically generated

Figure 2.3 Generalized Logs

Text

Description automatically generated

Figure 2.4 Removed Log

For our PCAP data, we needed to use only the columns necessary for the machine to use to understand which is infected with WannaCry. We decided to use the info and protocol. Since the rest of the columns are different to each run its best to use the columns that are consistent.

For the info column, the IP addresses were different, so we decided to generalize them by using regex to fill each IP address with \*’s.

Table

Description automatically generated

Figure 2.5 Normal PCAP CSV

Table

Description automatically generated

Figure 2.6 Starred PCAP CSV

Classifiers & TF-IDF

We decided to use three classifiers from sklearn, Extra Trees, Gradient Boost, and Support Vector Machine. Since classifiers can’t take strings as input, we used TF-IDF to see which term is the most relevant term in all our documents.

Table

Description automatically generated

Figure 3.1 TF-IDF Pre-Encryption CSV

Table

Description automatically generated

Figure 3.2 TF-IDF PCAP CSV

Predictions & Thresholds (Tenfold)

To give us accurate results, we used a Tenfold or K fold technique, this technique involves saving a bucket of data to use for testing and using the rest of the buckets to train the classifier, then repeating x number of times to get a consistent score.

Diagram

Description automatically generated

Figure 4.1 Tenfold Setup

Performance

We tested by retraining and averaging their scores by x number of iterations. The score is how accurate the classifier was at predicting the truth. Score = (TP + TN) / (TP + TN + FP + FN).

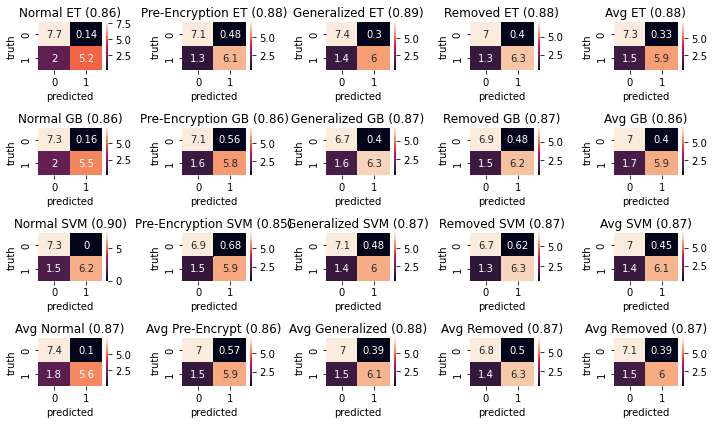


Figure 5.1 Confusion Matrix of Cuckoo Logs

Text

Description automatically generated

Figure 5.2 Ascending Order of the Score in 5.1

From multiple iterations, Extra Trees is the best out of the other classifiers and having the full logs past encryption help with accuracy.

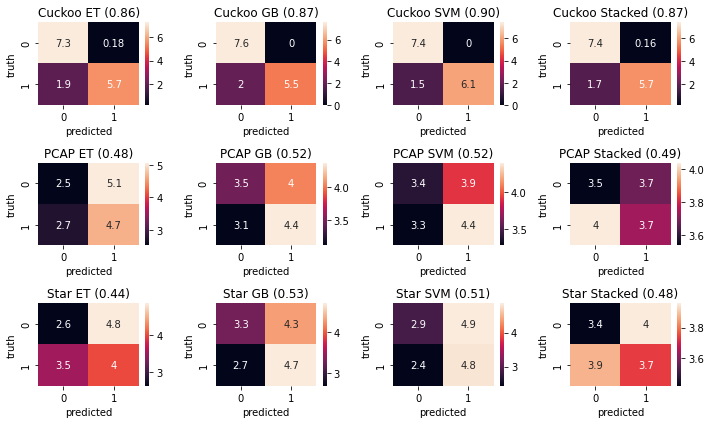


Figure 5.3 Confusion Matrix of all Logs

Graphical user interface, text

Description automatically generated

Figure 5.4 Ascending Order of the Score in 5.3

\*Figure 5.3 and 5.4 has the full logs, meaning it has pre and post encryption.

Since we want to warn the user when they have a virus before it starts encrypting Pre-Encryption is needed.   
We also created a python function to input a single scenario to give a probability and if it passes a certain probability then it is deemed either infected or benign.

Text

Description automatically generated

Figure 5.5 Test of Probability

In Figure 5.5, the models are trained then tested in one scenario, for the above the classifiers they are both the Extra Trees, since they were the best from previous runs, then combined and average to produce a result of it being infected with a 76% percent chance and with a truth of one the algorithm was right.

References

[1] https://cuckoosandbox.org

[2] https://pyinstaller.org/en/stable/

[3] https://cuckoo.readthedocs.io/en/latest/installation/host/requirements/

[4]