Detection of ransomware through behavioral analysis

Mohammad Ghaemi
Mentored by Joshua Bennett, Jeremy Rubin, Gerald Williams, Carl Quincy, and Mason McGlasson

Introduction

In 2016, the CryptoWall ransomware alone was estimated to have cost the United States 325 million dollars in damages (Sgandurra, Muñoz-González, Mohsen, & Lupu, 2016). Ransomware is a type of malware that restricts access to files, forcing victims to pay a ransom. As we rely more on the internet, we become vulnerable to these attacks. Currently, most antivirus companies detect ransomware by making a specific ID for different malware based on their code (signature-based). Antivirus solutions that only use signature-based detection cannot detect new ransomware until a new ID is constructed. Today's malware often change code slightly with every computer infected, preventing detection. One study showed that "out of the 3,000 unique analyzed exploit kits […] only 6% of their signatures were found in VirusTotal" (Sgandurra, Muñoz-González, Mohsen, & Lupu, 2016). New approaches analyze the behavior of files to detect unknown malware. One study by Chen and Bridges (2017) discovered patterns in ransomware behavior by analyzing their actions and finding features most unique to them with a Term Frequency-Inverse Document Frequency (TF-IDF), an algorithm that could find behaviors most unique to ransomware. A second study created a machine learning algorithm (which detects patterns through trial and error) trained on labeling files as ransomware and goodware by analyzing the file actions after being run (Sgandurra, Muñoz-González, Mohsen, & Lupu, 2016). This project will combine a TF-IDF and machine learning algorithm for a more accurate final product which will be able to detect ransomware types it has not analyzed.

Materials and Methods

The ransomware files were taken from the VirusShare database and the benign files were taken from the System32 folder of a Windows 7 computer. Cuckoo Sandbox, a program that gathers behavior data by executing files, was then set up on a laptop running Linux. Cuckoo Sandbox analyzed each file and outputted a report file for each. These report files had to be manually examined as Cuckoo Sandbox did not always record information from ransomware. All code in this project was written in Python. A list of Dynamic-link Libraries (DLLs) and Application Programming Interfaces (APIs) files used and the number of times it used them were extracted automatically from report files. APIs and DLLs that referenced encryption were removed because once ransomware encrypt files, the detection is useless. The extracted information from the ransomware was combined into one list while the benign files were placed in separate lists.

Materials and Methods (cont.)

A set of DLL/API lists were randomly separated to be used for training and the remaining were used for testing. The training data was used to calculate the TF-IDF value for each training file. The sum of all TF-IDF values a file used was calculated. If a certain threshold was met, then that file was predicted to be ransomware. All the DLLs/APIs a file could use were assigned index values in a list. A copy of this list was made for every file and when a file used a DLL/API, the TF-IDF value was added or subtracted from the appropriate element (based on whether the TF-IDF was unique to malware or benign files). This data was used to train a neural network with a hidden layer of size 437 (where the dot product of the hidden layer and the input list is taken) to predict the file as malicious or benign (all steps are shown in Figure 1). Each cell in the hidden layer had a probability of 0.4 to do no computations (called drop out) and override old information with new information at a rate of 0.001 (learning rate).

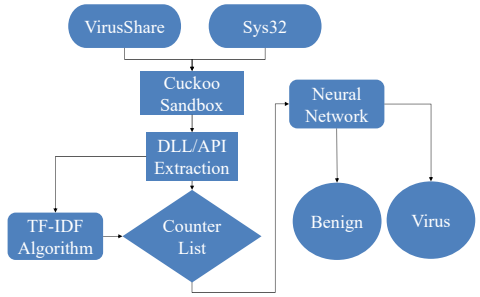
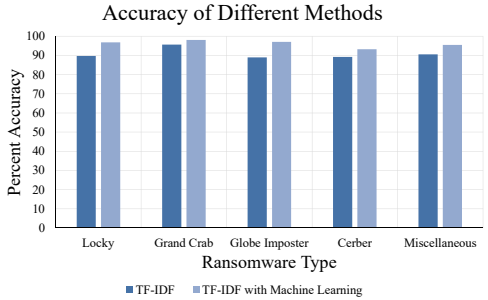


Figure 1 (left): Flowchart of the steps it taken to gather and extract DLL and API data from a set of files using Cuckoo Sandbox and combine a TF-IDF algorithm and a neural network to detect whether the files are malicious or benign. More information on the neural network's parameters can be found in the text above.

Results



Graph 1 (above): Compares how well the TF-IDF algorithm and the machine learning with the TF-IDF algorithm detected ransomware. The TF-IDF category had a median accuracy of 89.66%, the "TF-IDF with Machine Learning" category had a median accuracy of 96.76%. A total of 453 malicious files and 453 benign files were used.

Accuracy without Training	
Locky	93.68
Grand Crab	93.47
Globe Imposter	88.40
Cerber	84.18
Miscellaneous	91.31

Table 1 (above): Displays the accuracy of the method that used machine learning with TF-IDF when testing files infected by ransomware types not previously trained against. The algorithm detected Locky the most and Cerber least.

Results (cont.)

The 1-Sample Sign test rejected the null hypothesis of there being no significant difference between the medians of the group that used both machine learning and the TF-IDF algorithm (*Med* = 96.11%) versus the group that only used the TF-IDF algorithm (*Med* = 92.39%), $p = 0.031$.

Conclusions

This project's intentions were to create an algorithm that relied on behavioral data taken from files when executed in order to detect whether the files were ransomware or benign. More specifically, the project attempted to combine two different algorithms to detect ransomware at a higher accuracy as well as detect ransomware types it had no data on, something static signature-based detection methods are incapable of doing. The project succeeded in detecting ransomware types it had never trained on before since it was able to detect them at a median accuracy of 91.31% (Table 1). The project also found that utilizing both the machine learning and TF-IDF technique made a statistically significant difference when compared to only using the TF-IDF technique (Graph 1). Given the nature of neural networks, further optimizations on hyper parameters such as the loss function and the network architecture can be done to increase the accuracy of the detection method. Currently, the project only uses DLLs and APIs of files, while other behavioral data still exist, the most significant and potentially useful being networking data. Further more, using a machine learning technique that focuses on order (like a recurrent neural network) can boost the accuracy greatly if the order in ransomware's actions follow a consistent pattern. The methods used in this project can be implemented into an online service that accepts files and outputs whether they are malicious or benign to the user's computer. Finally, techniques in this project can and should be expanded to detect all malware types due to the spread of polymorphing code which allows malware to bypass the current common static detection methods utilized by antivirus vendors.

References

Chen, Q., & Bridges, R. A. (2017). Automated behavioral analysis of malware a case study of WannaCry ransomware. Retrieved from https://arxiv.org/abs/1709.08753

Sgandurra, D., Muñoz-González, L., Mohsen, R., & Lupu, E. C. (2016). Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. Retrieved from https://arxiv.org/abs/1609.03020SS