Volume: 8 Issue: V Month of publication: May 2020
DOI: http://doi.org/10.22214/ijraset.2020.5325

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.429
Volume 8 Issue V May 2020- Available at www.ijraset.com

Ransomware Protection Tool based on Recurrent Neural Network (RNN)

Nandhini S[1], Antony John Martin[2], Siddharth Srivastava[3], Mohammad Musfik[4]
[1]Associate Professor, [2, 3, 4]Student, Department of Computer Science and Engineering, SRMIST, Ramapuram, Chennai

Abstract: Ransomware is emerging as a leading cybersecurity threat to both organizations and individuals. According to Bitdefender "Ransomware is a form of malicious software (or malware) that, once it's taken over your computer, threatens you with harm, usually by denying you access to your data.
The invader strains a ransom from the target, to reestablish contact to the data upon expense". Ransomware can be mainly categorized into two, Locker ransomware which locks the computer or device and Crypto ransomware, which will prevent the user from accessing their files by encrypting them.
A network-based intrusion detection system was implemented, employing two independent classifiers working in parallel on different levels: packet and flow levels.
This proposed system is built on the deep learning approaches used in malware discovery for distinguishing ransomware from imitation arrangements. We extant specific recurrent neural networks for catching resident happening designs in ransomware orders using the idea of attention mechanisms.
Keywords: Locker Ransomware, Cybersecurity, Malware *(key words)*

I. INTRODUCTION

Ransomware attacks are becoming a serious cyber threat to organizations and individuals around the globe. Unlike traditional malware, ransomware attackers hijack the system and demand money to reverse the attack. A recent study, reports an immense increase and steady growth in new ransomware samples and attacks. In order to counteract and detect malware it was created a variety of methods implemented in the framework of the antivirus software. At first, detection was based on static analysis of compiled signatures, identifying malicious program file.
At the moment, with the active use of the creators of malware cloaking techniques from detection such as encryption, polymorphism, metamorphism, obfuscation and others, it is not enough to use only signature-based detection. Another important imperfection may be the increasing with the time signature database and deceleration check for malware.
Often, new variants of malware are not distinguished from their predecessors due to the limitations of classification systems relying only on static analysis. Thus, techniques like content-based, signature-based and pattern matching techniques for malware analysis are becoming less effective to detect and classify new variants of ransomware and provide insight information about the threat, goals and behaviors of ransomware.
The proposed system improves neural cell to integrate attention in knowledge from the ransomware arrangements, known as ARI (Attended Recent Inputs). The ARI cell, while dispensation the input sequence, also learns from a recent history in the form of a subsequence. It studies consideration weights consistent to each current input and uses their conforming implication when dispensation the input.
ARI is the capability to assimilate current input consideration inside the bigger cell operating directly on a sequence. By providing an unspoken contribution consideration, we give assistance by utilizing the courtesy weights at each timestep in computing both the concealed initiation from the cubicle, as well as the lockup memory. The planned scheme achieve a detailed investigation of ransomware executables in demand to recognize mechanical belongings that can be subjugated by machine learning schemes. We recognize an presence of minor reiterating designs within extended arrangements of ransomware possibly conforming to recurrent encryption processes.
We extant a new recurrent neural network constituent for manipulating the reiterating patterns by including attention mechanisms on the ideas of a arrangement knowledge module. We extant an LSTM modified of our cell called ARI-LSTM.Through experiential outcomes on a ransomware dataset, we demonstrate that ARI-LSTM achieves meaningfully healthier than an LSTM for the chore of ransomware discovery.

II. RELATED WORK

In this section we review some of the techniques/approaches used for malware classification. Traditional approaches use signature-based and pattern matching techniques to identify and classify malware. Often, new variants of malware are not distinguished from their predecessors due to the limitations of classification systems relying only on static analysis. [3] Olga Hachinyan [1] applied proactive technologies that use the testing of program for the presence of certain features, often taking place in the malware. For proactive methods usually do not require big signature database, and they are also capable of detecting more unknown malware, however, often have a greater number of false positives.

To deal with increasingly intensifying quantitatively and qualitatively malware it is necessary to develop proactive detection methods. This article is focused on proactive methods based on API calls analysis and proposes a new method with the use of multiple sequence equation to identify common signs of malware. Krzysztof Cabaj [2] proposes a real-life malware evolution observed for a long time.

Their impact on both toolsets and methods are presented based on practical development of systems for malware analyses and new features for existing tools. Jill Slay [5] UNSW-NB15 is compared with KDDCUP99 data set by considering some key features and it shows the benefits. Mustafa Kaiili [4] proposes a system analysis in which the network traffic of Locky ransomware to extract a number of informative network features.

Locky often spreads over spam emails that contain malicious attachments in the form of macro-enabled office documents. This attachment runs a script, called a downloader, to download Locky's executable file from a URL and install it on the victim's system. When fully installed, Locky tries to find and contact its C&C server(s) to exchange encryption keys and execute the intended malicious actions via one of the following strategies: It uses an encrypted list of hardcoded IP addresses to establish a TCP session with its C&C server(s).

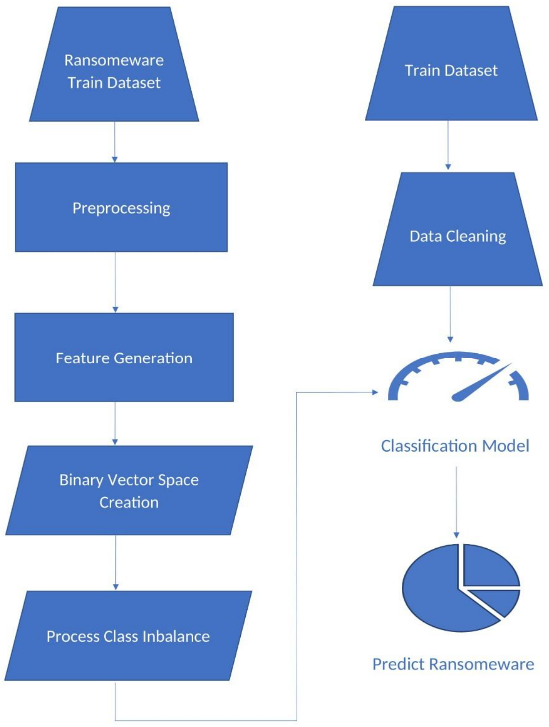
If these hardcoded addresses are all unreachable, i.e., blacklisted, or the session was disrupted, Locky tries to find its C&C server(s) by running the Domain Generation Algorithm (DGA) that periodically generates a large number of pseudo-random domain names. Locky keeps sending DNS requests about these names until the actual C&C server(s) is found. If the DGA method also fails, Locky uses the NetBIOS Name Service (NBNS) protocol hoping to find a previously infected machine on the local network that has already resolved the C&C server name.

III. FRAMEWORK

The framework will work in four different modules. The first module includes local static information and second module is data pre-processing A detection algorithm based only on local static parameters can detect malware before it runs. An effective algorithm based only on local static parameters is the most effective and avoids any loss of user data. The whole dataset is loaded as the comma separated value file (CSV file). Afterward which the information pre- processing is completed. Since we have finished the de-duplication our data necessitates some preprocessing earlier, we go on more with breakdown and making the forecast model. Henceforth in the Preprocessing stage we do the subsequent in the order below:

A. Start by eliminating the html tags
B. Eliminate any punctuations or incomplete set of distinct characters like, or. or # etc.
C. Verify if the term is made up of English letters and is not alpha-numeric
D. Verify to see if the length of the word is greater than 2 (as it was researched that there is no adjective in 2-letters)
E. Convert the word to lowercase
F. Remove Stop words
G. Finally Snowball Stemming the word (it was observed to be better than Porter Stemming)
H. After which we collect the words used to describe positive and negative reviews

After preprocessing the data will go to part where dimensionality reduction is done and which is then used for classification where it is carried out through a supervised approach, in which the system is trained with samples whose label (i.e., benign, generic malware or ransomware) is known. Such method has been used in preceding works with outstanding results [3, 2, 8]. Thus, the classification model is used to predict the Ransomware.



IV. SYSTEM ARCHITECTURE

A. *Local Static Information*

A detection algorithm based only on local static parameters can detect malware before it runs. An effective algorithm based only on local static parameters is the most effective and avoids any loss of user data. A common technique used in commercial antivirus software is to obtain the local static parameters from the analysis of the program binary. However, some ransomware strains use code obfuscation techniques or a polymorphic behavior which hinders detection. The static information obtained from the files is related to text strings or function calls.

TEXT STRINGS Common strings found in ransomware binaries are ‘‘ransom’’, ‘‘bitcoin’’, or ‘‘encrypt’’. It can also contain well-known domain names or IP addresses. The anti-malware software can search for keywords or set phrases. It is usually complemented with a deeper analysis of static or dynamic parameters because the method is prone to false positive alerts.

FUNCTION CALLS The most common function calls found in ransomware programs are related to cryptography algorithms (key generation, encryption, and decryption) and file access. Binary inspection can detect the use of these suspect function calls. They can be functions from well-known dynamic system libraries or statically linked libraries.

B. *Pre-Processing*

In this stage, the request is examined to excerpt its data. The mandatory statistics is pulled out by only reviewing the executable code and does not achieve any examination on other essentials, such as the application Manifest. Feature Abstraction (System API). In this stage, the code outlines established from the preceding stage are added and examined to excerpt the linked System API material (either packages, classes, or methods). The incidence of such parts of evidence is then calculated, thus manufacturing a trajectory of statistics (feature vector) that is directed to a classifier.

C. *Dimentionality Reduction*

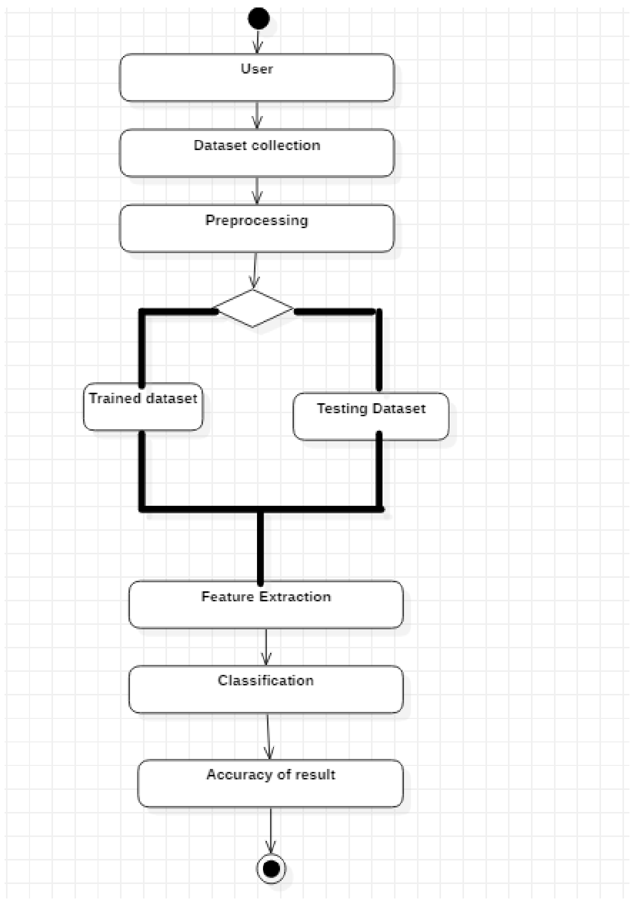
To understand how instruction opcodes, contribute to ransomware detection, we employed dimensionality reduction techniques through PCA. We observed that 6 percent of the principle components explain 50 percent of the variance across the full feature set. Moreover, we found that 99 out of a hundred of the modification is enlightened by 60 out of a hundred of the principle mechanisms. Additional testing showed that we could significantly reduce the feature count without compromising the accuracy of the model. For example, in the random forest model we can achieve the same accuracy of 99.87 percent with less than 3 percent of the features.

D. Classification

Classification is approved out over a supervised approach, in which the system is accomplished with examples whose tag (i.e., benign, generic malware or ransomware) is recognized. Such method has been used in preceding works with outstanding outcomes [3, 2, 8]. Our methods hire Random Forest classifiers, which are particularly valuable to handle multiclass difficulties, and which are extensively used for malware discovery. The complication of such classifier's rests on the number of trees that comprise them. Such a quantity must be enhanced throughout the drill phase.

V. IMPLEMENTATION

Implementation is the stage in the project where the theoretical design is turned into a working system and is giving confidence on the new system for the users, which will work effectively and efficiently. It involves careful planning, investigation of the current system, its constraints on implementation, design of methods to achieve the change-over, and evaluation of the change-over methods. Apart from planning, the major task of preparing the implementation is the training of users to get accustomed to the aesthetics of the system. The more complex being implemented, the more involved will be the system analysis and the design effort required just for the implementation.

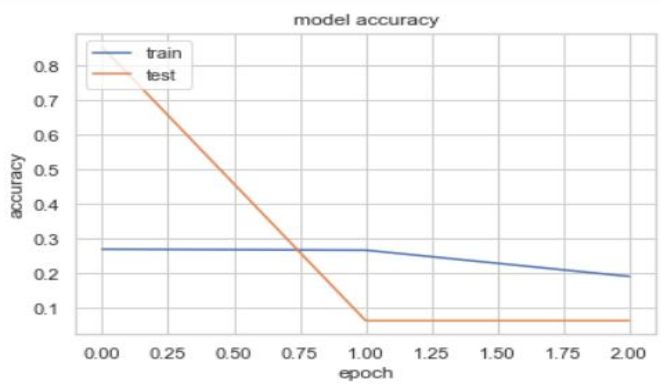


VI. RESULT

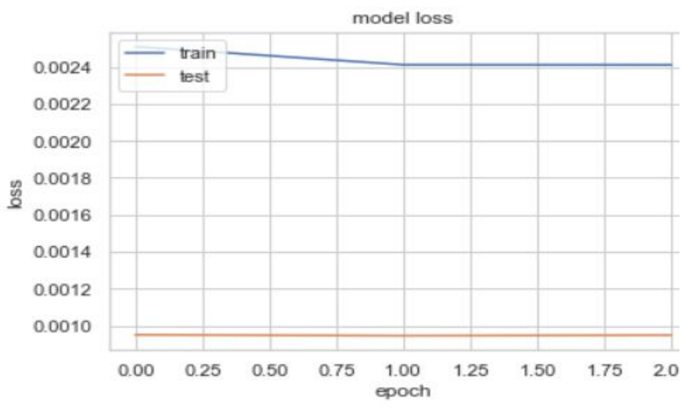
This scheme will classify a presence of minor reiterating designs within long classifications of ransomware possibly conforming to recurrent encryption processes. We extant a new recurrent neural network constituent for manipulating the repeating patterns by incorporating attention mechanisms on the ideas of an arrangement learning component.

This plan uses an LSTM irregular of our cell called ARI-LSTM. With experiential results on a ransomware dataset, we show that ARI-LSTM performs suggestively healthier than an LSTM for the chore of ransomware discovery. With the ARI cell, we extant a method for including attention at the contributions of a preparation, which can be used by glitches penetrating to relations within recent inputs. We are using two datasets i.e., (1) Non-Ransomware Dataset and (2) Ransomware Dataset. We test both the datasets and predict the range of the ransomware using the Sequential model. These predicted ranges can be used to determine whether a ransomware is detected or not.

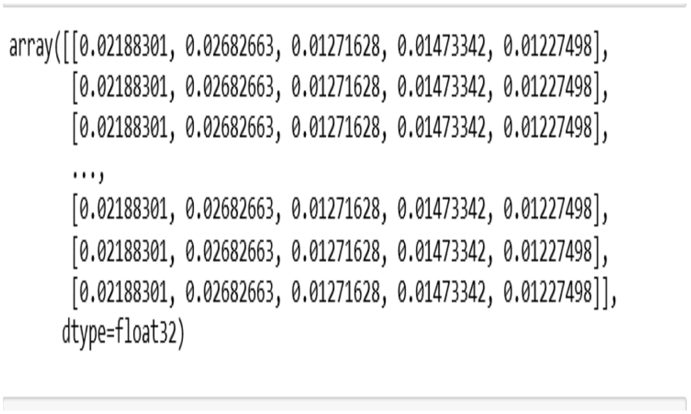
Plotting a graph between the train and test data against Accuracy vs Epoch to get the Model Accuracy:



Plotting a graph between the train and test data against Loss vs Epoch to get the Model Loss :



The predicted range for the Ransomware dataset which can be used as a reference for the detection of the ransomware:



```
array([[0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498],
       [0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498],
       [0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498],
       ...,
       [0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498],
       [0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498],
       [0.02188301, 0.02682663, 0.01271628, 0.01473342, 0.01227498]],
      dtype=float32)
```

VII. CONCLUSION

This system will identify an existence of small repeating patterns within long sequences of ransomware potentially corresponding to repeated encryption operations. We present a novel recurrent neural network component for exploiting the repeating patterns by incorporating attention mechanisms on the inputs of a sequence learning module. This system propose an LSTM variant of our cell called ARI-LSTM . With empirical results on a ransomware dataset, we show that ARI-LSTM performs significantly better than an LSTM for the task of ransomware detection. With the ARI cell, we present an approach for incorporating attention at the inputs of a sequence, which can be used by problems sensitive to relations within recent inputs.

REFERENCES

[1] "Malware detection and subscriber protection infographic," Alcatel-Lucent, Tech. Rep., Access Date 10Oct,2015.[Online].Available:https://www.alcatellucent.com/solutions/security-guardian-infographic
[2] S. Young and D. Aitel, The hacker's handbook: the strategy behind breaking into and defending networks. CRC Press, 2003.
[3] K. Timm, "Strategies to reduce false positives and false negatives in nids," Tech.Rep., Access Date 10 Oct,2015.[Online].Available:http://www.symantec.com/connect/articles/strategies-reduce-false-positives-and-false-negatives-nids
[4] K. Julisch and M. Dacier, "Mining intrusion detection alarms for actionable knowledge," in Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining - KDD02. Association for Computing Machinery (ACM), 2002.
[5] M. Tavallaee, E. Bagheri, W. Lu, and A.-A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in Proceedings of the Second IEEE Symposium on Computational Intelligence for Security and Defense Applications 2009, 2009.
[6] J. McHugh, "Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory," ACM transactions on Information and system Security, vol. 3, no. 4, pp. 262–294, 2000.
[7] A. Zaalouk et al., "OrchSec: An Orchestrator-Based Architecture for Enhancing Network-Security Using Network Monitoring and SDN Control Functions," Proc. Network Operations and Management Symp., 2014, pp. 1–9.
[8] R. Jin and B. Wang, "Malware Detection for Mobile Devices Using Software-Defined Networking," Proc. GENI Research and Educational Experiment Wksp., 2013, pp. 81–88.
[9] Chandrasekar R. "Malware Detection using Windows API Sequence and Machine Learning". Pondicherry Engineering College, 2012, 5 pages.
[10] Microsoft MSDN: Windows API Index. [Online]. Available: https://msdn.microsoft.com/ruru/library/windows/desktop/ff818516%28v=vs.85%29.aspx.
[11] A. Sung, J. Xu, P. Chavez and S. Mukkamala, "Static Analyzer of Vicious Executables (SAVE)", 20th Annual Computer Security Applications Conference.
[12] "Anubis", Anubis.iseclab.org, 2018. [Online]. Available: http://anubis.iseclab.org/. [Accessed: 18-Feb-2018].
[13] "Cuckoo Sandbox - Automated Malware Analysis", Cuckoosandbox.org, 2018. [Online]. Available: https://cuckoosandbox.org/. [Accessed: 18-Feb-2018].
[14] J. T. Juwono, C. Lim and A. Erwin, "A Comparative Study of Behavior Analysis Sandboxes in Malware Detection," in International Conference on New Media (CONMEDIA), 2015.
[15] K. Rieck, P. Trinius, C. Willems and T. Holz, "Automatic analysis of malware behavior using machine learning", Journal of Computer Security, vol. 19, no. 4, pp. 639-668, 2011.
[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann and I. Witten, "The WEKA data mining software", ACM SIGKDD Explorations Newsletter, vol. 11, no. 1, p. 10, 2009.
[17] T. Sochor and M. Zuzcak, Study of Internet Threats and Attack Methods Using Honeypots and Honeynets. Cham: Springer International Publishing, 2014, pp.118–127.[Online].Available:http://dx.doi.org/10.1007/978-3-319-07941-7_12
[18] P. Baecher, M. Koetter, T. Holz, M. Dornseif, and F. Freiling, The Nepenthes Platform: An Efficient Approach to Collect Malware. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 165–184. [Online]. Available: http://dx.doi.org/10.1007/11856214_9
[19] M. Xu, L. Wu, S. Qi, J. Xu, H. Zhang, Y. Ren, and N. Zheng, "A similarity metric method of obfuscated malware using function-call graph," Journal of Computer Virology and Hacking Techniques, vol. 9, no. 1, pp. 35–47, 2013. [Online]. Available: http://dx.doi.org/10.1007/s11416-012-0175-y
[20] C. Guarnieri and A. Tanasi. malwr.com website. [Online]. Available: http://malwr.cm
[21] M. Vasilescu, L. Gheorghe, and N. Tapus, "Practical malware analysis based on sandboxing," in 2014 RoEduNet Conference 13th Edition: Networking in Education and Research Joint Event RENAM 8th Conference, Sept 2014, pp. 1–6.
[22] K. Cabaj, P. Gawkowski, K. Grochowski, and A. Kosik, "Developing malware evaluation infrastructure," in Proceedings of the 2016 Federated Conference on Computer Science and Information Systems, ser. Annals of Computer Science and Information Systems, M. Ganzha, L. A. Maciaszek, and M. Paprzycki, Eds., vol. 5. IEEE, 2016, pp. 1001–1009.
[23] K. Cabaj, P. Gawkowski, K. Grochowski, and D. Osojca, "Network activity analysis of cryptowall ransomware," Przegla̧d Elektrotechniczny, vol. 91, no. 11, pp. 201–204, 2015.