RansomedLSTM: Ransomware Detection using Recurrent Neural Networks

Muhammad Tayyab
Richard Stretanski
Jason Carlton
College of Engineering and Computer Science
University of Michigan - Dearborn
Email: tayyab,rstretan & jcarlto@umich.edu

***Abstract*—Cyber criminals have been leveraging the cryptography to launch a specific kind of denial of service attack against the users, where they encrypt and hold the victims data. Which renders the victim unable to use the data until the ransom is paid. Such attacks have proven to be devastating, resulting in a loss of billions of dollars every year. The most alarming situation is the one where the attackers use it as a denial of service attack tool rendering the services based infrastructure unusable. One of the most offensive attacks of this kind in recent times is WannaCry which was launched in May 2017. That infected over 230000 computers including the United Kingdom's National Health Service. The attackers demanded the ransomware in the bitcoins. The only kill-switch which was found by the researchers was to block a web domain address found in the reverse engineering. Breaking the encryption of the ciphers depends on the strength of the cipher suites. However, the attackers use the standard crypto libraries for encryption which means, the resulting cipher is powerful enough and cant be broken easily. The detection and the defense against the ransomwares is relatively a new topic in the research community, though few solutions have been provided which are unfeasible and come with the inherent deficiencies. In this paper, we have proposed a novel idea for the detection and prevention of the ransomwares leveraging the power of deep learning. We trained a Long Short Term Memory (LSTM) Neural Network, a variant of Recursive Neural Networks, to detect the ransomware infection. We gathered the binaries infected with the real world ransomwares and the programs which perform the encryption and decryption legitimately. The trained LSTM is able to classify the ransom-wares accurately. In the paper we have also discussed the implementation of the system in the realtime machines at instruction level by harnessing the power of the Application Specific chips designed specifically for the deep learning which overcomes the performance overhear and does not impact the overall performane of the system. We have compared our results with the other results from the literature already and analyzed the accuracy of our system.**

***Index Terms*—Ransomwares, CryptoWall, WannaCry, Petya, Recursive Neural Networks, Long Short Term Memory Networks, Hardware based security.**

I. Introduction

With every new morning, a new kind of cyberattack is making its way to the news, posing the serious threats to the wide range of internet users ranging from general public to the governments. The attacks have been evolving and the strategies have become tricky enough to leave the researchers in barren land full of questions with no leads to answers. One of such hard problems and threatening attacks, the new kind of malicious programs have emerged in recent three or four years. This cyber disease is known as the ransomware. The attackers exploit the vulnerable machines just like the other traditional malwares but encrypt the victim's machine. They encrypt the important documents and desktop, leaving the user clueless. They demand the user ransom to make them useable again.

It may looks like a science fiction but it is true and the just eight months ago a ransomware named as WannaCry made the international community cry like its name. Within three days it infected more than 230000 computers world wide. This figure includes the general public computers as well as the critical infrastructure based computers as well. The United Kingdom's National Health Services computers were vulnerable at that time and were shut down. The system has to run on emergency basis to keep its operation going. WannaCry exploited a vulnerability in the Windows based machine known as EternalBlue in the SMB protocol. This vulnerability got long discussion and have the political aspect as well which is out of scope for this paper. There was no direct breakdown of the WannaCry other than to patch the systems. No detection at time. The only killswith was to shutdown the domain found in the reverse engineering.

Then shortafter WannaCry, another infamous threat appeared on the horizon of crypto attacks. This time it targeted the critical infrastructure precisely and lead the infamous Chyrnobell Nuclear Power plant to shutdown temporarirly [2]. Even in the most recent months, October 2017, there is another threat identified in the Eastern Europe. This threat is named as Bad Rabbit [1]. This is not surprising because strong crypto libraries are out there deployed into the realworld systems which makes the encryption super strong. That is a good thing. This creates the problem when the viruses use these implementations for the malicious intents and now we can not recover the encrypted files. Obviously to circumvent this problem, we can not go for weakening the security. It is the need of the hour to go for the out of box solution. Researchers has shown a great deal of interest already and tried to solve the dilenma, however the proposed solutions have inherent deficiencies or overhead. The proposed systems may solve the problems in present but to fight for the future threats should be in the vision of security community as well.

We believe for most of the security researchers or even the common people the reported incidents above are sufficient inspiration to work in this domain. At least we can say this statement to ourselves. In this paper we have proposed a novel approach to fight the epidemic of ransomwares where we are leveraging the power of deep learning. We have proposed and showed a proof of concept based on the Long Short Term Memory Networks which are a kind of Recurrent Neural Networks. We trained the system on the binaries which were infected with the ransomwares and were able to distinguish with great deal of accuracy. We have proposed the integration into the system at architecture level as well with some possibilities to leverage the power of neural network hardware accelerators. Our final goal is to turn this research into the detection and the prevention of the ransom-wares at instruction level by the hardware.

In the next section we will be talking about the related work and the rest of the paper is organized in this fashion. In Section III the architecture of our system and the overview of LSTM is given. In Section IV we have discussed the methodology and later in Section V, the results of our research are analyzed. Then the paper is concluded with the conclusion section and the future work respectively.

II. Related Work

One of the breakthroughs in the area of ransomware detection is undoubtedly done by the researchers in [**?**]. The system keeps track of the operation performed by a certain process on the given block of the data. A typical ransomware will read the data, encrypt it and then delete the original data chunk. This pattern can be used to classify the ransomware and the legitimate process. Because, the legitimate processes usually do not delete the chunk of the data after encryption. The challenge here is to identify the encryption in the data. However, this can be achieved by monitoring the entropy of the data at I/O. A low entropy in the Input and the high entropy at the Output shows the encryption. This is an intersting solution implemented at the filesystem level. However it can easily be broken by the ransomwares. It is easy to bypass the patter of the access. One more thing it just detects and may be by that time the damage has been occured. In the paper the authors have proposed to use the desktop background changes to detect the ransomware operation. This approach is not enticing and looks like it is kind of borrowed from the human user detection. Like common users only know when the damage has been done by seeing a message in our desktop.

Some conventional efforts have also been made like to have a cloud back up of the whole filesystem [4]. This is a trivial solution and can easily be evaded if the ransomware encrypts the backup. It is a costly solution in terms of the networking and whole new infrastructure. In ideal case we can see as replicating the current machines into new machines. The problem is still there. We just have made the other bite available when the current meal is poisonous. It is a good temporary solution but the problem here is to stop current meal from becoming poisonous.

An in depth and elegant solution which focuses at the recovery once the damage has been done is in [5]. As most of the ransomwares use the standard libraries, shared or static for encryption, they proposed to start keeping track of the session keys and store them in a secure vault. If the system is hit with the ransomware, we can go to vault and recover the encrypted files. This solution has inherent defficiency that if the static libraries are obfuscated and can not be hooked up there is no way to retreive the session keys. If the customized secure implementation is used by the authors of ransomwares, this system is unable to perform there.

Authors in [**?**] proposed a system to prevent malicious programs from accessing the files to be encrypted. Implemented at the kernel, Redemption works to mediate access to the file system. If Redemption receives a request to access files in a specific path from an application, Redemptions monitor is called. Meanwhile, Redemption creates a corresponding file in the protected area and handles the write requests. A malice score is given to the process which is then compared to a pre-configured threshold. One limitation of this system is that every request for access is first submitted to Windows I/O allowing attackers exploit the message system by creating scenarios that generate fake alert messages accusing any program of being ransomware and rendering this solution untrustworthy to users, resulting in the user turning off Redemption. In this scenario, usefulness would depend on users knowing which programs are trustworthy.

Another limitation of Redemption is how malice score is calculated. A ransomware could feasibly remain below the malice score threshold by performing selective content overwrite, using a low entropy payload for content overwrite, or launching periodic file destruction. Each of these things on their own is not enough to pass the malice score threshold, however it would only result in a single file being encrypted and compromised at a time. Using the same exploit an attacker can also skip the encryption of the users files and just lock the desktop once the ransomware is installed. This is not nearly as difficult to overcome for the user as encrypted files, however it may trip up some users.

Keeping in view of the weaknesses of most of these works, we proposed a solution which addresses these problems properly. Our solution is re-configurable and tends to be better with the time because it is a deep learning based solution, which can learn from the use itself if deployed. The basic architecture of the solution is given in next section.

III. Architecture

Given the weaknesses and limitations of the previous works, we are proposing a novel idea for the ransom-ware detection leveraging the power of deep learning neural networks. Once a neural network is trained, it can be implemented in the prefetcher between the memory and the processor. Once implemented in prefetcher, it can detect the malicious content in the real-time manner before the execution. We have opted the Recursive Neural Networks: Long Short Term Memory (LSTM), for the detection of the ransom-wares.

The biggest challenge for opting the LSTM is the training phase. Conventionally the training of the LSTM takes a lot of time. Before getting the maximum performance, we trained the networks over 20 times and each iteration took on average 4 hours. The details of the challenges and the lessons are listed in the later sections. Before discussing the results and the implementations a brief overview for the LSTM is given in the next subsection.

A. Recursive Neural Networks: LSTM

Many of the real-world problems specially the speech recognition, the movie plot understanding and most commonly the text prediction in the emails or sms texts in the smart mobiles, all of these applications naturally require the recent past knowledge i.e. the information is purely sequential. It is impossible to solve these problems based on the current input solely and require the information realted to the past states. Conventionally the simple neural networks are not capable of storing the past information and most importantly understanding the relationship with the current states of the network. These problems are solved by the Recursive Neural Networks.[8]

As the name implies that the recursive neural networks take the previously computed results as well as the current input to compute the present output. In other words, the RNNs builds a relationship between the current and the past outputs. This ability has been leveraged to solve the above aforementioned sequential problems.

There is another bottleneck with the RNNs is that how deep in the past the present output depends. Some problems may depend only a few previous values while some may be dependant upon the ouput values long ago in past. This is the problem which is somehow addressed in the Long Short Term Memory Networks. The detailed working overview of an LSTM is out of scope of this paper however we are just describing the overview of the architecture and the reason to opt this solution over the other potential solutions.

LSTM networks are trained using the back propagation through time approach, that is the error and the states are retained within the time domain as well. Unlike the conventional neural networks the LSTMs have the memory blocks which are connected though layers. These memory blocks are smarter than the normal memory blocks in terms of information retaining, access and the store. All of these operations are controlled by the triggering factors in analogue manner implemented through the simple neurons. This is a beautiful structure which is mesmerizing as well as somehow difficult to understand. This property of the memory blocks, cells, help the LSTMs to retain and process the information smartly across the time steps.

The operations on a given cell are managed through the gates. There are three type of gates inside the given block: Forget Gate, Input Gate and Output Gate. These gates decide based on the input sequence to decide that whether the information should be retained, accessed or updated. The cells are considered the building block of the LSTM layer. Given these abilities, the LSTMs are considered to be Turing complete in the theory, so they can mimic all of the mathematical operations. This lead us to chose the LSTM. The network which we trained for this paper have the below structure in terms of layers:

- Sequence Input Layer
- LSTM Layer
- Fully Connected Layer
- Soft-max Layer
- Classification Layer

We trained the network for different sizes of the LSTM layer to maximize the performance and accuracy of the classification.

In the next section we will be briefly discussing the enviornment setup.

IV. Methodology

A. Platform Setup:

Setting up the target platform and the building platform is basic block in any project's requirements. While starting this project, initially Linux based systems were considered as the potential target or victim machines for the ransomwares. However, the ransom-ware binaries for the Linux based systems are not very common and hard to get. Based on this observation, we changed our target platform to Windows based machines.

For developing and training the neural networks, the ideal platform is python based development given that the libraries for machine learning are very powerful along with the performance capabilities. We tried to setup the python based platform along with the tensor-flow. The unfamiliarity of the authors with the machine learning libraries in python became a factor in context of the short time for the project deadline. One of the authors had prior experience of working in the MATLAB's machine learning environment, the priority was given to MATLAB. This saved us a great deal of time and kept us on track. Though the MATLAB could not leverage the performance of the computing hardware for the LSTM training, it took more time, however the results at the end were good. We are able to prove the proof of concept with this platform. Now it can easily by shifted to the python for the further full scale implementation.

For the project MATLAB's Neural Network and Deep Learning Toolbox are used primarily. In the next section we're describing the data-set we have used for the project.[9]

B. Dataset:

Ransomware Dataset: As described in the previous section that due to insufficient ransomware infected binaries binaries for the Linux based platforms, we changed our target platform to Windows. It was though that the binaries for the windows based platform will be easy to get. However it was an ardous task to get the dataset for the ransomware infected binaries. So, getting the dataset became the main bottleneck of the project. As opposed to the leading papers in the literature, we do not have the big data set to test upons becuase of the limited

scope. We tried to contact the authors of the UNVEIL and the PAYBREAK with no success. From one of the authors of the PAYBREAK we got the hashes of the binaries they used, it was found later that the public access is denied as well.

Upon searching through the public resources like github we found a good dataset for the viruses overall and luckily there were 14 famous ransom-wares available and some of them with different versions [7]. A list of the used ransomwares is given below:

- WannaCry
- CryptoWall
- Locky
- Jigsaw
- Mamba
- Matsnu
- Petya
- Radamant
- TeslaCry
- Rex

As this list contains the most dangerous families of the ransom-wares in recent times, it is satisfactory to interpret the results. Obviously, the results could have been quite convincing if there were more binaries infected with these ransom-wares. To avoid the damage in real life, we did not try to infect the binaries ourselves. However, for the future, this is could be a potential option to go for to get more binaries given the full control over the resources to circumvent the damages.

Simple Encryption-Decryption Dataset: The binaries which perform simple encryption and decryption are easy to find. However, these should be standalone. For example, Skype does a lot of encryption and decryption but it is distributed into many binaries. So, as a quick solution we gathered the Slueth Kit's binaries which perform such operations being standalone. These are close to 20 binaries and makes a good dataset for the proof of concept.

In the next section we are going to describe the training process adopted:

C. Training:

The binary files are being read in the raw form in 64-bit numbers. The whole dataset is divided into two classes the ransomewares are given class label 1 and the simple binaries are given the class label 2.

The files have a broad range in terms of the size. Some binary files are few KBs while others are in MBs. If the LSTM is trained on this raw setting it will start truncating or padding and may result in loss of huge semantics underlying. So, to solve this problem, the raw data of these files is sorted in the ascending order based on the size of file. This makes the files of similar size together in a minibatch. In the mini-batches now the similar sized files are placed and the truncating or padding will not impact the semantics of the data.

A LSTM network is developed based on the layers described above. For the finalized network, the size of the layers is given below:

- Sequence Input Layer: 1 (size of each input)

```
>> Ransomewars
Training on single CPU.
|=========================================================================|
|  Epoch  |  Iteration  | Time Elapsed | Mini-batch | Mini-batch | Base Learning|
|         |             |  (seconds)   |    Loss    |  Accuracy  |    Rate      |
|=========================================================================|
|       1 |           1 |         6.07 |     0.6931 |   100.00% |       0.0100 |
|       5 |          50 |      1007.80 |     0.5279 |   100.00% |       0.0100 |
|      10 |         100 |      2063.06 |     0.3766 |   100.00% |       0.0100 |
|      15 |         150 |      3069.39 |     0.3232 |   100.00% |       0.0100 |
|      20 |         200 |      4117.98 |     0.3001 |   100.00% |       0.0100 |
|      25 |         250 |      5211.72 |     0.2883 |   100.00% |       0.0100 |
|      30 |         300 |      6380.54 |     0.2811 |   100.00% |       0.0100 |
|      35 |         350 |      7489.80 |     0.2758 |   100.00% |       0.0100 |
|      40 |         400 |      8516.34 |     0.2728 |   100.00% |       0.0100 |
|      45 |         450 |      9411.78 |     0.2764 |   100.00% |       0.0100 |
|      50 |         500 |     10267.90 |     0.2809 |   100.00% |       0.0100 |
|=========================================================================|
```

Fig. 1. Training of LSTM. Smaller mini-batches resulted in the 100% classification of the ransomwares.

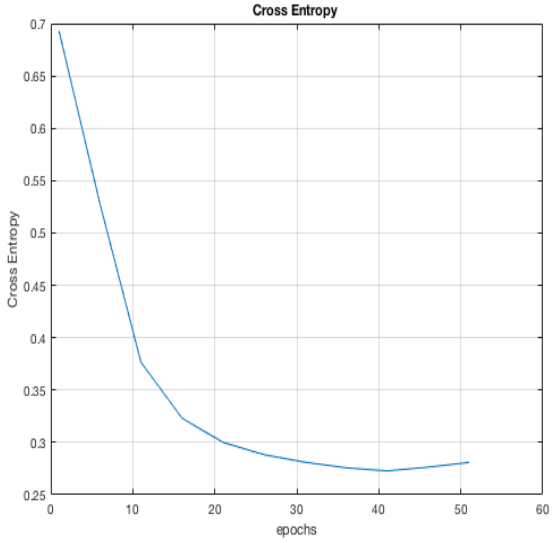

Fig. 2. Training of LSTM. Cross Entropy decay with the epochs.

- LSTM Layer : 50
- Fully Connected Layer:2 (number of classes)
- Soft-max Layer
- Classification Layer

For training, the mini-batch size is finalized to be 2 binary. For which the LSTM was trained with 100% accuracy. The training algorithm used is SGDM with the max epochs 50. The training performance and the time taken to train is discussed in the upcoming section.

V. Results:

In the Fig.1, we can see the classifier training results. The cross entropy has decreased from the start while having the

TABLE I
COMPARISON OF PAYBREAK AND RANSOMEDLSTM

Family	PAYBREAK	RansomedLSTM
WannaCry	-	Y
CryptoWall	Y	Y
Locky	Y	Y
Jigsaw	-	Y
Mamba	-	Y
TeslaCrypt	N	Y

classification performance 100%. The training of the LSTM initially was a strenuous task given the big minibatch sizes. The reason could be because of the different binary sizes there could have been the loss of semantics of the raw data due to padding and truncating. This is the reason when we decreased the sizes of the minibatches, the training performance increased and finally upto 100%.

It is a big success to classify the lethal ransomwares' infected binaries. A comparison of the performance with the other systems is given below:

In the Table. I, we have compared the performance of both of the system on common ransomware families. Because of the difference in datasets we can not get much information about the comparison, however we can see that TeslaCrypt was not detected by the PAYBREAK however our system, RansomedLSTM can detect it.

Our system compares better when it is compared with the other solutions like PAYBREAK and the UNVEIL. Because the limitations are there for both of the systems which have already been discussed in the literature review section of the paper. Our system is not measuring the entropy which can be evaded easily by the ransomwares by coming up with the format preserving encryption. Or in case of the PAYBREAK, if the attacker statically links the crypto libraries which obfuscating the implementations it becomes hard for PAYBREAK to detect the malicious encryption activities. RansomedLSTM is independent of these limitations because of its Turning Complete nature and virtually executes the given payload to make a decision before even the binary instructions are fed to the processor. The question of the realtime implementation of Ransomeed LSTM is resolved in the next section.

A. System Integration:

To implement this solution in the realtime, one of the possible places to integrate this solution is in the prefetcher between the memory and the processor. To implement at hardware level, we can leverage the power of the ASICs or the custom developed chips. One such example could be the Intel's NNP which is a coprocessor specially designed for the depp learning. It will not impact the efficiency of the system overall. There will be another advantage is that in case of the performance of the NN deteriorates, we can reprogram the neural network.

Another potential solution could be the FPGA for NN. It sits between the memory and the processor. In case of any malicious intent we can simply halt the processor to execute.

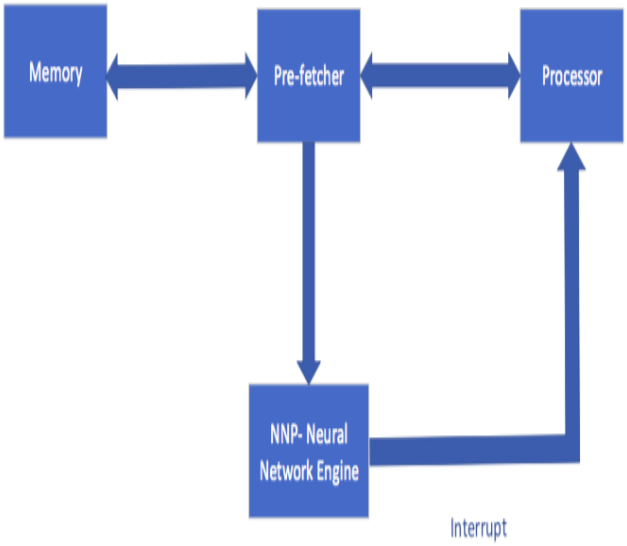

Fig. 3. Implementation overview of the system

The overall abstract implementation is given in the figure. 3. The NNP block can be replaced by the hardware accelerator chips or customized FPGA based chips. This block is configurable and can run the updated LSTMs. We can increase the performance by continuously training the NN as well. This solution will not impact the efficiency of the system overall. The computational load will be on the NN engine or the chip. If the malicious binary or instructions are found we can interrupt the processor and stop the system from executing it.

It can be think of a costly solution, however the cost is nothing as compared to the loss possible. Spending few bucks per system should not be a problem as compared to giving millions in ransom and bearing the consequences.

VI. CONCLUSION:

As a proof of concept, this project is a success and a milestone for the bigger picture. Now we can easily move to next steps towards the real world implementation. This project has been a great learning opportunity overall. We started from scratch and within short span trained a system with high accuracy to detect the ransomwares.

We explored the implementation in the realtime as well proposing a hardware accelerator for NN. It is predictable that the products will be shipped with the NN chips in the near future just like the crypto engines are being shipped to accelrate the computations. Our soultion is harnessing that capability to save the load on the main processor itself while offloading the burden to a coprocessor chip. It is a potential solution which can be scene as the detection as well as the prevention at the same time. We don't have to worry about the recovery when we are just stopping the malicious binaries to execute. There is no need to maintain a secure vault or loading the system with potetial solutions. We are not claiming the best possible solution, we just explored a possible solution

which seems to be efficient and viable. There are many other bottlenecks which are yet to be addressed before claiming the final version. These hurdles are briefly described in the next section.

VII. Future Work:

The current presented work is a proof of concept. There are many things which are needed to be done. The most important thing to do is the collection of bigger data-set. Specially the binaries which are infected with the broad range of the ransomwares.

The development should be ported to the python environment given the robustness of the python and the ability to harness the hardware of developing machine. Thus generated neural networks can be ported to the simulators. Thus the performance of the solutions can be analyzed before proceeding to the final implementation at hardware level.

The final step is to implement at the hardware level. We can use on board computers running Windows OS so we can customize the hardware ourselves. Integrate NNP like chips to accelerate the Neural Network implementation giving the chip to control the execution at instruction level such that the processor can be interrupted to stop the execution of the instructions.

VIII. Contributions:

A. Muhammad Tayyab:

Contributed in the proposal writing, the literature review. In the implementation phase, setup platforms and tested them, worked in collecting and finding the binaries. Studied the LSTM in detail, implemented and trained the system in MATLAB. Finally wrote the Corresponding sections of the report.

B. Richard:

Contributed in the proposal writing and proposal presentation . Richard worked in the data collection part. He was involved in the Literature review and was responsible for the introduction and the literature review section in the final paper.

C. Jason Carlton:

Jason contributed in the proposal presentation and worked in the platform setup phase of the project.

References

[1] https://www.kaspersky.com/blog/bad-rabbit-ransomware/19887/

[2] https://www.kaspersky.com/blog/expetr-for-b2b/17343/

[3] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson,and Engin Kirda. 2016. *UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware.* In 25th USENIX Security Symposium (USENIX Security 16). USENIX Association, Austin, TX, 757772.

[4] Vijayan Prabhakaran, Andrea C Arpaci-Dusseau, and Remzi H Arpaci-Dusseau. 2005. *Analysis and Evolution of Journaling File Systems.*. In USENIX Annual Technical Conference, General Track. 105120.

[5] Eugene Kolodenker, William Koch, Gianluca Stringhini, and Manuel Egele *PayBreak : Defense Against Cryptographic Ransomware*

[6] Amin Kharraz and Engin Kirda *Redemption: Real-time Protection Against Ransomware at End-Hosts*

[7] http://github.com/ytisf/theZoo/tree/master/malware

[8] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[9] https://www.mathworks.com/help/nnet/examples/classify-sequence-data-using-lstm-networks.html

[10] H. Kopka and P. W. Daly, *A Guide to LaTeX*, 3rd ed. Harlow, England: Addison-Wesley, 1999.