Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2019) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

Complex Adaptive Systems Conference with Theme:
Leveraging AI and Machine Learning for Societal Challenges, CAS 2019

Detecting Ransomware Using Process Behavior Analysis

Abdullahi Arabo [1,*], Remi Dijoux [1,2], Timothee Poulain [1,2], Gregoire Chevalier [1,2]

[1]Computer Science Research centre, The University of the West of England, CSRC, Bristol, UK, BS10 5PD *abdullahi.arabo@uwe.ac.uk
[2]Institue Universitaire de Technologie De La Reunion, France

Abstract: Ransomware attacks are one of the biggest and attractive threats in cyber security today. Anti-virus software's are often inefficient against zero-day malware and ransomware attacks, important network infections could result in a large amount of data loss. Such attacks are also becoming more dynamic and able to change their signatures – hence creating an arms race situation. This study investigates the relationship between a process behavior and its nature, in order to determine whether it is ransomware or not. The paper aim is to see if using this method will help the evading malicious software's and use as a self-defense mechanism using machine learning that emulates the human immune system. The analysis was conducted on 7 ransomware, 41 benign software, and 34 malware samples. The results show that we are able to distinguish between ransomware and benign applications, with a low false-positive and false-negative rate.

Keywords: Ransomware, malware, cyber security, machine learning

1. Introduction

Ransomware is a category of malicious software which, when run, disables the functionality of a computer, or encrypts the files in it. Ransomware operates in many different ways, from simply locking the desktop of the infected computer to encrypting all of its files [3]. The ransomware program displays a message that demands payment to restore functionality or decrypt files. The malware, in effect, holds the computer at ransom. This is highly profitable, with 2.9 percent of compromised users paying out. An investigation into one of the smaller players in this scam identified 68,000 compromised computers in just one month, which could have resulted in victims being defrauded of up to $400,000 USD [2]. Ransomware is significantly grown since the last decade and it has a financially motivated cyberattack. Europol has reported in 2018 that ransomware is one of the biggest threat in cyber security. In order to test a viable solution, we wrote a Python program, to perform two main functionalities:

- Collect data about resources used by a process, by using a server-oriented solution;
- Detect and kill any suspicious process.

The data collected are used to train a machine learning system, which might detect ransomware itself after enough training. The discrimination could be made in two ways: with machine learning, or by defining pre-set thresholds. The latter is used to analyze data which cannot fit into the machine learning system, like string variables. Malware is malicious software created in order to harm a computing system. The main goal is generally to earn money, by extracting critical information to sell them later or blackmailing the targeted user. A brief definition of terms is provided in Table 1

Table 1Key Terms Definitions

Virus	Viruses are able to be replicate and they can be expanded on the network in order to infect other computers
Worm	Instead of the virus, the worm is able to be spread autonomously on the network
Trojan	A Trojan is a software that looks seemingly legitimate but in reality it contains a malicious program
Ransomware	Ransomware is a malicious software who encrypt personal data in order to take our computer in hostage. If you want to restore your data, you need to pay a ransom

When the user executes an infected file, the ransomware will ask to the attacker server an encryption key. The attacker server will send back the encrypted key to the ransomware. The key contains a very long character string which is unique to the victim PC.

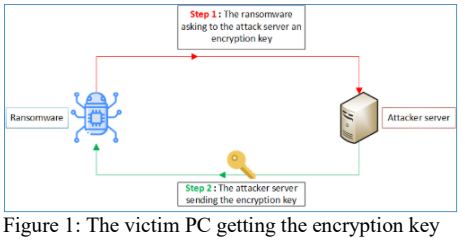

Figure 1: The victim PC getting the encryption key

Figure 2: The victim PC being encrypted

Then, the ransomware will start encrypting all personal data and asking you a ransom in order to recuperate it. All your data will no longer be accessible. The pictures on the test machine have been encrypted, and now have the extension ".WNCRY ". For our proposed architecture, we had two virtual machines. One virtual machine running on Windows 7 and the other virtual machine is a Linux machine running on Ubuntu 16.04 LTS. All virtual machines were on a private network in order to avoid propagation into the network. The Linux machine IP address is 192.168.56.104 and the windows machine IP address is 192.168.56.103.

2 Related work

The days of worrying with the issue of phishing emails from a proposed wealthy individual or those with deceased family members who are trying to transfer their wealth to other countries and similar email are phasing out gradually. The most lucrative aspect now is in the form of Ransomware which can also be spread via a similar concept i.e. phishing emails. The security industries have been able to develop ways of mitigating the first scenario via the identification of typos and grammatical errors in such email but finding it difficult to help prevent ransomware attacks. These attacks are more lucrative and successful, as a result when you are infected your option is either to pay off, lose your data or in some other instances infect a given number of others systems before your data will be decrypted. Having said that, there is a silent movement in terms of a project nomorerandome (https://www.nomoreransom.org/) which helps solve/minimize the problem of paying for ransom. In this section, we review existing work in order to perform a state of the art related to ransomware detection.

An overview of ransomware attacker and countermeasures has been provided by tailor Jinal P and Patel Ashish D [11] where comparisons of recent studies, key contributions, and limitations were presented. Detailed "CryptoLocker" analysis and explanation has been provided by *Ross Brewer et al.* [**7**], this also includes the details on how to handle a ransomware attack. For example one of the common method to make a backup of all your data on the computer. As well as advising organizations to develop an incident response team. This will help to ensure a prompt response in a scenario where time is of the essence to stop or contain a serious situation.

R. Richardson et al. [**9**] insists on the fact that human is the key factor in most of the ransomware infection, by

affirming that 59% of infections came by e-mail (phishing, infected attachment), and 24% came via websites. *K. Cabaj et al.* [**4**] are proposing a solution based on ransomware traffic characteristics, more precisely the HTTP requests made by the infected host to the remote server in order to exchange the asymmetric encryption key. By analyzing first a large range of ransomware (359 CryptoWall samples), they were able to perform detection rates of 97-98%. *A. Azmoodeh et al.* [**5**] proposed a solution based on the ransomware's power consumption fingerprint. They were able to achieve a precision rate of 89.19% and a detection rate of 95.65%. A cloud-based solution, called Malware Detection as A Service (MDaaS) has been presented by *A. Bhardwaj et al.* [**1**]. The system is composed of three virtual environments running services for Malware Behavioural Analysis, Malware Code Analysis, and Malware Reporting. Despite being a real-time anti-malware, the files analysis might take a lot of time and resources, involving high latency and delay in case of poor internet connection. The use of three virtual environments instead of one means 3 times more resources, which is a disadvantage compared to the standard cloud service anti-malware solution. If the user is offline, his device cannot be protected and he could get infected by a worm using an offline propagation method, like through removable drives. An alternative solution composed of three important parts: the Crypto function hooking, the key vault, and a file recovery system has been demonstrated by *E. Kolodenker et al.* [**8**]. They are combined to form a cohesive system that is able to reverse file encryption performed by hybrid cryptosystem ransomware. *Song et al* [10] have proposed a process monitoring ransomware solution for android devices by focusing on some preventative measures based on process usage, memory usage input and output rates to detect abnormal behaviors of processes. Our proposed solution is an extension to this basic concept by looking more into key process usages in terms of disk usage, reads and writes as well as creating a multi-threaded approach to increase the efficiency of the detection and prevention process. We have also demonstrated our solution based on a larger dataset for both ransomware and benign ware and malware samples. Our solution also includes the analysis of DLL's used as well as API calls, which provides a more compressive and detailed analysis of the process to provide a dynamic preventative solution.

3 Proposed detection method

The final goal is to determine whether a process running within an ecosystem is ransomware or not, in order to stop it. To achieve that, the following points are checked:

- Which DLL APIs are called? How much the system resources are used? Which files are opened by the process?

Once the program got that information, it uses manually entered pre-sets coupled to a machine learning solution in order to determine the nature of the process. The purpose of our paper is to propose a solution able to detect and neutralize ransomware. To achieve that, we have to collect information about systems behavior, by executing them on a virtual machine. Nonetheless, if a Python script is running on this machine, there is a high chance that the script and all it generates might be destroyed. So once the program retrieves any information about a process, it sends the data directly to a monitor using a secure connection channel, the monitoring system is Linux based to minimize potential ransomware infection. We executed ransomware, malware and trusted software's on the windows machine. The python program is able to transfer the data we collect from ransomware, malware and trusted software to the Ubuntu machine.

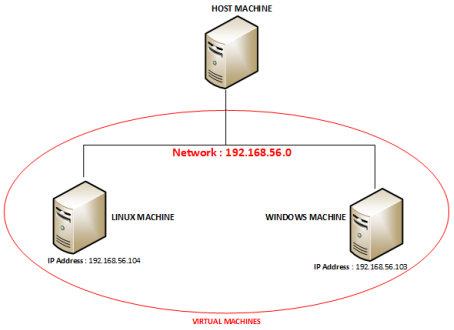

Figure 3: Lab architecture

For the reason that not all the data we retrieve needs to be periodically monitored, we initially created two different threads:

1. The "analysis" thread which collects the variable data and loop until the process is ended;

2. The “info” thread which collects the constant data, send them and stop.

And because all the data are not taking the same amount of time to obtain them, we created two other threads for the “slow-to-collect” data: The “tagfile” thread, reading the tagfile content; the “files” thread which collect the name of all the files opened by the process.

The flowchart in Figure 4 represents how the main program works. It shows the different type of data in each thread. By doing this type of separation, it optimizes the program, because obtaining one value may take time, during which time the thread is paused. With this system, the constants are returned once, and the variables are returned periodically following the process execution speed.

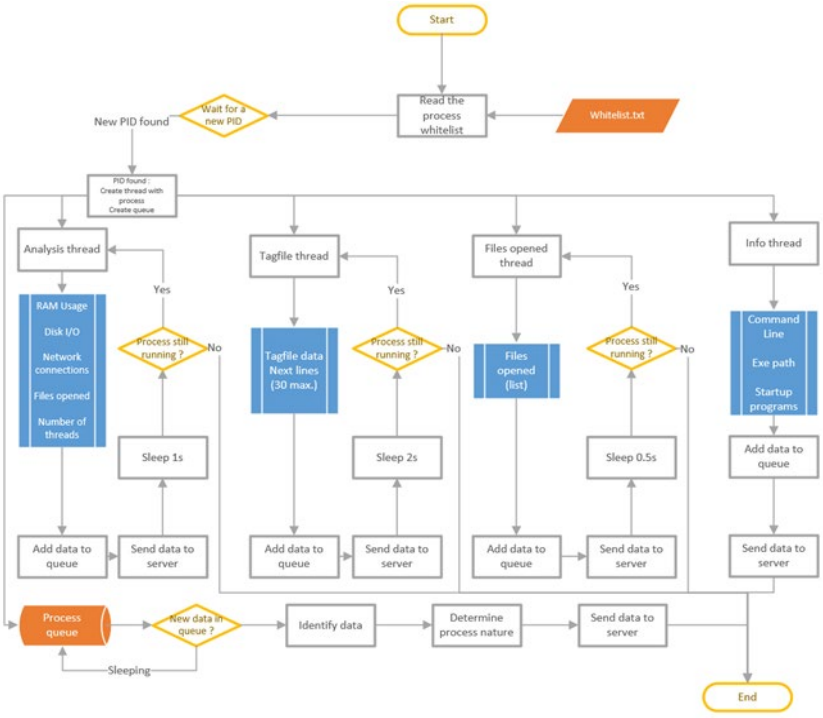

Figure 4: Main program flowchart

We first thought about collecting any possible data, so we could determine which one might be useful. At the end of our paper, we are not using all of the data in Table 2.

Thread	Data	Frequency
Analysis	process name	Every 1s
Analysis	PID	Every 1s
Analysis	CPU Usage	Every 1s
Analysis	RAM Usage	Every 1s
Analysis	Disk usage	Every 1s
Analysis	Network connections	Every 1s
Tagfile	DLLs used (API calls)	Every 2s
Info	PID	One-Time
Info	Shell command line	One-Time
Info	Executable path	One-Time
Info	Startup programs	One-Time
Files opened	Files opened	Every 1s

Table 2: Data distribution among the threads

All the collected data are added to a queue before being sent to the server. In this way, the main thread associated with the analyzed process can gather the data from the different threads, and proceed with the discrimination using all the data collected.

We used third-party software in order to strengthen our ransomware detection program: Tiny tracer, which allows us to gather information about DLLs API calls and also capturing the transition between sections of the traced module. Tiny tracer is a program made in C++. When you execute a program with tiny tracer he will generate a ".tag" file which contains all the DLLs API calls of the program. This tag file can be easily read with a simple notepad.

We analyzed 41 benign applications, especially software installers because they are the ones with the closest behavior to ransomware. The reason for this is the number of disk inputs and outputs made by these types of program, which are way higher than others.

We analyzed 34 malware processes. They were useful for our final detection program: by doing that, we could see if there is a difference between a malware's behaviors and any other type of software.

We analyzed different ransomware (7 in total). We will describe the behavior of some of them, in order to manually identify patterns.

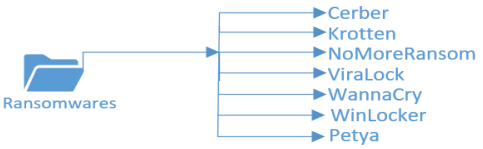

Figure 5: Ransomware analyzed

After analyzing ViraLock (Figures 6, and 7), we saw that the CPU had a lot of variation. The maximum value of the CPU is 25% usage. On the other side, the memory usage is more stable. The maximum value is 2% usage. The CPU usage can help determine the presence of a malicious program, but might not be the key factor. The values of disk read and write is drastically increasing. The maximum value of disk write is around 450000. The maximum value of disk read is around 420,000. This is the most important factor among all the values we analyzed. The values of bytes read and writing is also increasing. The maximum value of reading is around 6,000,000,000. The maximum value writing value is around 50,000,000,000. Those values are extremely high, in contrast to benign applications.

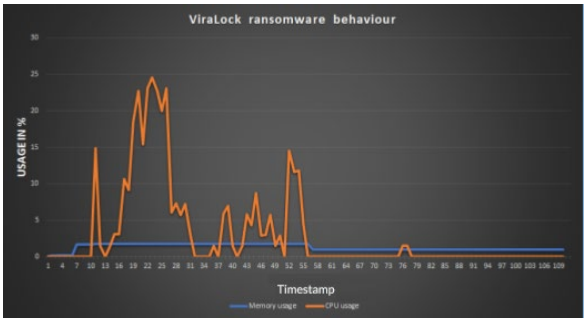

Figure 6: ViraLock CPU and memory usage

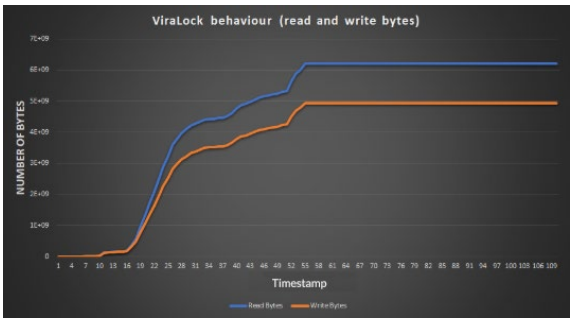

Figure 7: ViraLock read and write bytes

When it comes to WannaCry, our Python program analyze it only four times before crashing. WannaCry ransomware is more powerful than the others. The maximum CPU used is 24.7% and the memory usage is very low (around 0.2% usage). The number of reading and write counts is increasing drastically. The maximum value of reading is 97 and it's 764 to write (pretty low). The maximum value of reading bytes is around 60 000 000. The maximum value of write bytes is around 120,000,000 (Figure 8 and 9 respectively).

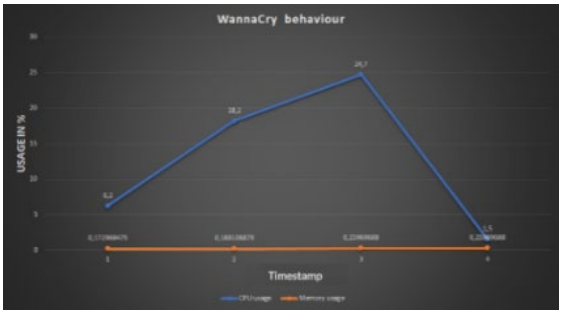

Figure 8: WannaCry CPU and memory usage

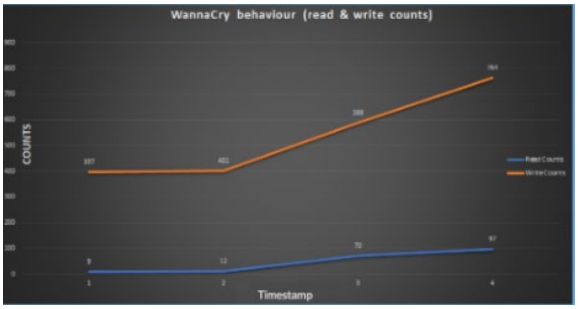

Figure 9: WannaCry read and write bytes

After analyzing Cerber (Figure 10) we saw that the maximum CPU usage is 37.5% and the memory usage is around 0.1%. We can suppose that Cerber is starting the process of encryption when the CPU is at 37.5%. The maximum read is around 2,200 and to write, it's 1,800. After 22 timestamp our python program was killed by Cerber. The number of reading and write is high. That is proof that the ransomware is active and is encrypting our data. The number of reading and write has been increased drastically. It is another proof of ransomware activity.

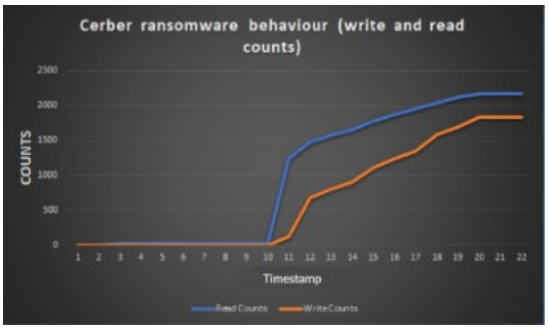

Figure 10: Cerber CPU and memory usage

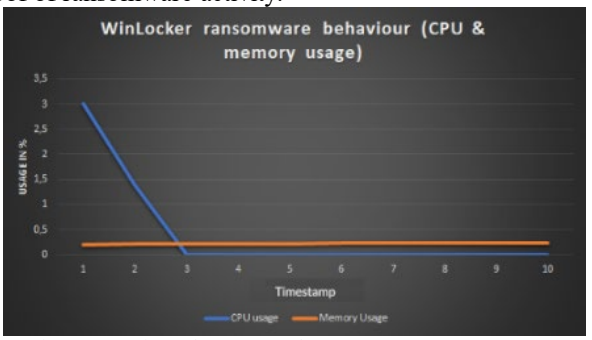

Figure 11: WinLocker CPU and memory usage

Winlocker killed the python program. The program analyses 10 times the process before crashing. The maximum value of the CPU is at process startup and it is 3%. The CPU is increasing at 1.5%. The memory usage is around 0.2%. As we can see (Figure 11), the number of reading and write counts are increasing over time. The maximum read count is 42 and the maximum write count is 20. We saw the number of bytes written and read is increasing as well. We improved graphs by classifying PID in order to have a good overview of ransomware behavior.

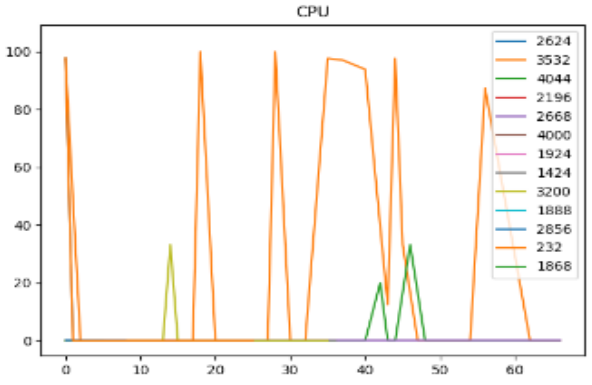

Figure 12: CPU usage

The data discrimination is divided into two parts, depending on the type of data to analyze. The reason for this is that we cannot put any type of data in the machine learning program, it supports only integers and floats. The strings will be analyzed using a threshold system and another method, as in Table 3.

Table 3 Discinimation Data

Discrimination method	Data	var type
Machine Learning	PID	int
Machine Learning	CPU Usage	float
Machine Learning	RAM Usage	float
Machine Learning	Disk usage	float
Threshold	API calls	String
Threshold	Files opened	String

We created a system to determine whether a data is a clue to spot ransomware or not: We use a weighted average system. For each data added, the values are analyzed by specific functions which contain thresholds. Then, the specific analysis functions return a value between 0 and 1, which can be compared to a mark, and an integer which is a coefficient. The value of this coefficient change following the threshold triggered in the function. Finally, all those values are merged to calculate a weighted average. If this average is superior to 0.5, the process is suspected to be ransomware, and the user is warned with a pop-up, which provides the user with an option to kill it or not.

With the data we collected, we were able to make a solution against ransomware. Our python program will be launched at the PC start-up. When we execute ransomware, a window will appear and tell us that this program is suspicious. If we click on yes, it will immediately kill the process. For this example we ran WinLocker and we have five values in total: **([0, 5], [0, 4], [1, 11], [1, 3], [1, 2]).**

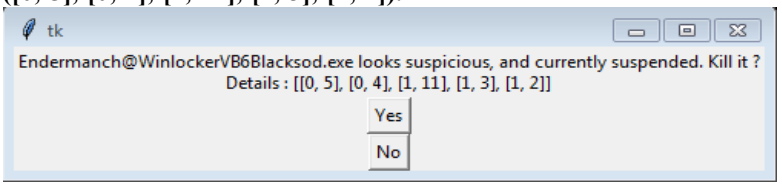

Figure 13: Windows displayed when ransomware is detected

For the first case of each value "1" mean it is detected as ransomware and "0" mean it is not detected as ransomware. The second value of each case corresponds to the coefficient used.

The first value corresponds to the machine learning detection. We choose "5" for the coefficient of machine learning. This coefficient will be the same whether it is detected as ransomware or not.

The second value corresponds to the files extensions. If the process found a file with an extension know as malicious, like "wnry" for example for WannaCry, the value will be "[1, 100]", so the coefficient will be at 100. If not, the value will be "[0, 4]", so with the lowest coefficient which will be "4".

The third value corresponds to API calls made by the DLLs. We collected all of the API calls of the ransomware DLLs's in order to find which function of a DLL is called. We classified those API calls and if a known function used by a DLL is called at least five times, the first case value will turn to 1. To calculate the coefficient, we count the number of iterative calls made by all DLLs functions and we divide by two.

$$Coefficient = \frac{Calls}{2}$$

The fourth value corresponds to disk usage. If a number of written bytes is higher than 100,000 and the number of written bytes is less than 5,000 we will have "[1, 10]" as output. If the number of bytes written is less than the number of reading bytes, then the output will be "[0, 5]" because we cannot have more read bytes than written bytes after our analyses. If the number of written bytes is higher than reading bytes, then the output will be "[1, 3]". And finally, if we do not have either of all the proposition, the output will be "[0.5, 1]".

Classifier	Neural Net	Nearest Neighbors	Linear SVM	RBF SVM	Gaussian Process	Decision Tree	Random Forest	AdaBoost	QDA	Naive Bayes
Accuracy	52.85%	57.06%	63.05%	60.50%	59.80%	70.14%	75.01%	61.64%	62.77%	58.55%

Figure 14 ML Experimentation Results

The last value corresponds to the threads. If the number of thread is less than 5, the output will be "[1, 2]". If the number of thread is higher than five the output will be "[0, 2]" because after analyzing the values collected, the number of a thread never went higher than five.

Making a graph with python was very effective to study the behavior of malware. With the system usage, we saw on the graph the variation of the different values. The CPU usage was an important factor to determine if there is a malicious process running on the computer. The write, read bytes and count on the disk can improve to detect ransomware. It allows us to have much important information which can be very effective in order to study a ransomware behavior.

The resulting output of process flags are saved as a CSV file and this is used as input to the machine learning module using a decision tree, random forest, and neural network. Out test result consists of splitting the dataset into training and test samples. The result as presented in Figure 14 shows that the random forest produces the best result and more need to be done to improve the results.

4 Conclusion

We were able to study a lot of different ransomware and extracting values from like the DLLs used and the system usage. We were able to increase my dataset of malware and ransomware. Even if the machine learning technique needs more training, we were able to implement a solution against ransomware attacks which allow us to detect the process and determine if it is ransomware or not with the API calls of each function used by DLLs, with the extensions, the disk usage and the number of threads.

Our system is able to detect zero-day ransomware attacks and warn the user about a potential threat. The benefit of our solution is that it does not need a signature database, but a dataset of ransomware and none- ransomware data. The more the dataset is enhanced, the more the system is successful in its discrimination. Our next step is to try and get this detection done within the first 5 seconds of malicious activity, then pass this information into an agent that will communicate this information securely to the ecosystem so as to form an early warning system for self-defense an create a more reactive preventative solution rather than a reactive defense – hence provide a zero-trust security solution.

Acknowledgment: This work has been supported via the Erasmus+ scheme as a collaboration between the University of the West of England and the Institue Universitaire de Technologie De La Reunion, France. It also forms part of the funded projection via the Grants4Growth scheme in collaboration with JOBIS IT Solutions.

References

[1] Akashdeep Bhardwaj, Dr. GVB Subrahmanyam, Dr. Vinay Avasthi, Dr. Hanumat Sastry (2016) Ransomware digital extortion: a rising new age threat, Indian Journal of Science and Technology, 9,14, 1-5 (2016)

[2] Gavin O'Gorman and Geoff McDonald (2012) Ransomware: A growing menace, Symantec Corporation

[3] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda (2016) UNVEIL: A Large-Scale, Automated Approach to Detecting Ransomware, 25th USENIX Security Symposium USENIX Security 16), (2016), 757–772, USENIX Association

[4] Krzysztof Cabaj, Marcin Gregorczyk and Wojciech Mazurczyk (2018) Software-defined networking-based crypto ransomware detection using HTTP traffic characteristics, Computers & Electrical Engineering, 66, 353–368, Elsevier

[5] Amin Azmoodeh, Ali Dehghantanha, Mauro Conti, Kim-Kwang Raymond Choo (2018) Detecting crypto-ransomware in IoT networks based on energy consumption footprint, Journal of Ambient Intelligence and Humanized Computing, 9(4), pp 1141–1152, Springer

[6] Akashdeep Bhardwaj, Dr. GVB Subrahmanyam, Dr. Vinay Avasthi, Dr. Hanumat Sastry (2016) Ransomware digital extortion: a rising new age threat, Indian Journal of Science and Technology, 9,14, 1-5 (2016)

[7] Ross Brewer (2016) Ransomware attacks: detection, prevention, and cure, Network Security, 9, 5–9, Elsevier

[8] Eugene Kolodenker, William Koch, Gianluca Stringhini and Manuel Egele (2017) PayBreak: defense against cryptographic ransomware, Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security, 599–611, (2017), ACM

[9] Richardson, Ronny, and North, Max M (2017) Ransomware: Evolution, mitigation, and prevention, International Management Review, 13, 1, 10, (2017)

[10] Sanggeun Song, Bongjoon Kim, and Sangjun Lee (2016). The Effective Ransomware Prevention Technique Using Process Monitoring on Android Platform. Mobile Information Systems, Volume 2016, Article ID 2946735, 9 pages, http://dx.doi.org/10.1155/2016/2946735

[11] Jinal P. Tailor and Ashish D. Patel (2017) A Comprehensive Survey: Ransomeware Attacks Prevention, Monitoring, and Damage Control, International Journal of Research and Scientific Innovation (IJRSI IV(VIS), pp 116-121