An I/O Request Packet (IRP) Driven Effective Ransomware Detection Scheme using Artificial Neural Network

Md. Ahsan Ayub*, Andrea Continella†, and Ambareen Siraj*
*Department of Computer Science, Tennessee Tech University, Cookeville, USA
†Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, NL
Emails: mayub42@students.tntech.edu, a.continella@utwente.nl, asiraj@tntech.edu

***Abstract*—In recent times, there has been a global surge of ransomware attacks targeted at industries of various types and sizes from retail to critical infrastructure. Ransomware researchers are constantly coming across new kinds of ransomware samples every day and discovering novel ransomware families out in the wild. To mitigate this ever-growing menace, academia and industry-based security researchers have been utilizing unique ways to defend against this type of cyber-attacks. I/O Request Packet (IRP), a low-level file system I/O log, is a newly found research paradigm for defense against ransomware that is being explored frequently. As such in this study, to learn granular level, actionable insights of ransomware behavior, we analyze the IRP logs of 272 ransomware samples belonging to 18 different ransomware families captured during individual execution. We further our analysis by building an effective Artificial Neural Network (ANN) structure for successful ransomware detection by learning the underlying patterns of the IRP logs. We evaluate the ANN model with three different experimental settings to prove the effectiveness of our approach. The model demonstrates outstanding performance in terms of accuracy, precision score, recall score, and F_1 score, *i.e.*, in the range of $99.7\% \pm 0.2\%$.**

***Index Terms*—Artificial Neural Network, I/O Monitoring, Malware, Ransomware**

I. Introduction

Ransomware is a special type of malware that can seize control over a computing system by exploiting vulnerability of email, remote desktop protocol, software, etc. Once under control, ransoms are demanded to return back the control to the owner. In general, ransomware can be categorized in two ways: locker - locking down the system to prevent any possible user actions / services; and crypto - encrypting important user data with a key to prevent any user access. This is however not a novel threat in the cyber space, as it was first reported in 1989 when a 20k floppy drive at an AIDS conference was infected and was named after as AIDS Trojan. Then, in 1996, Young and Yung [15] presented their intuition on how cryptography and its applications can be used for extortion-based threats. The paper demonstrated an information extortion attack, a type of cryptovirological attack, in which an author of the virus is able to ask the victim to get back the possession of the valuable pieces of information in exchange for the session key. Since then, ransomware defenders and researchers have been busy detecting as well as reporting new ransomware samples and families out in the wild very frequently, and it is becoming harder and harder to protect against this ever-changing enemy.

State-of-the-art Machine Learning (ML) techniques are being used as efficient detection schemes for diverse applications, such as, malware analysis [3], network-based intrusion detection system [12], etc. With the unique ability to differentiate the characteristics of benign and malicious actions, Supervised Learning in the domain of ML allows us to identify what is good and what is bad. The process of this learning starts by feeding a ML model input data with a label representing benign or malicious records. For many years, malware analysts and researchers have been leveraging this learning process with the use of system calls for improved malware detection or to discover new variance of malware suites [14]. One of the ways ransomware, a special type of malware suite, leaves its footprints of the damage to victim machine is through I/O Request Packet (IRP) logs[1]. In 2015, Kharraz *et al.* [6] described the usefulness of monitoring the file system for changes in the IRP logs to defend against ransomware which later got adapted by researchers in this domain as a viable ransomware detection technique. Our research takes detection using IRP logs further by taking into account the granular level activities spawned by each process triggered in the user space as a result of the interaction between the ransomware and the file system inside the kernel space. Additionally, it is centered around the IRP logs captured in the ransomware executions in presenting an effective detection scheme.

Problem Statement. As ransomware defenders and researchers gather IRP logs during ransomware executions and build state-of-the-art solutions to combat against this ever-evolving malware suite, ransomware, our study aims to devise an effective detection scheme by extracting actionable insights from granular activities at the process level as the encryption of the data assets are underway by the ransomware.

With the additional acquired knowledge of different users' behaviors (benign records), *i.e.*, home, office, and developer, we construct an effective ML model using Artificial Neural Network (also known as Neural Network) to discover and learn the underlying patterns of the benign as well as malicious IRP

[1]https://docs.microsoft.com/en-us/windows-hardware/drivers/gettingstarted/i-o-request-packets

logs. After completion of the learning process with training data, we evaluate the model's binary classification performance with testing instances and obtain outstanding results.

II. Background

A. I/O Request Packet (IRP)

The I/O Request Packet (IRP) is considered as a common method for requesting input/output (I/O) operations between the user and the kernel mode. When a user executes a command to open a file, the I/O system service inside the kernel mode carries out the task. At first, it looks up file names and checks access rights. Secondly, it locates the file inside the file system and allocates memory for the IRP request inside the I/O manager. Then, I/O manager passes the IRP information to the file system driver from where the task to open the file gets completed. At last, after receiving an I/O status from the IRP, the I/O manager frees the memory and returns a handle to the user with a success or failure operation status [2].

To describe the structure of IRP, there are three types of such operation: IRP; FIO (Fast I/O) – designed to directly transfer data between user buffer and system cache; and FSF (File System Filter) – designed to support IRP operations on file system[2]. The log contains several pieces of information regarding a process that triggers the operation, *e.g.*, process id, parent id, thread id, and process name. We observe, there are four special types of IRP flags: No Cache, Paging I/O, Synchronous API, and Synchronous Paging I/O. In addition, there are several other important flag attributes available in the IRP logs: Major Operation Type – signifies the type of IRP operation; Minor Operation Type – dependent on major operation type; and Status – designed to map the flag value to a human-readable format. Along with these flag values and timestamps of pre-operation as well as post-operation, the log captures some additional useful features too relating to file system, such as, File Object – responsible to provide all the file properties, File Name, Buffer Length, and Entropy.

For Windows XP and later operating systems, Microsoft provides a Minifilter Driver, a kernel-mode driver, that the third parties can hook into their tool and track down system calls made by the programs. This utility driver has been very popular among malware researchers in order to dissect malware, *e.g.*, ransomware, infection as they use it to collect and then analyze system calls (or IRP logs) [4].

B. Prior Work and Datasets Collection

We base our study on I/O Request Packet (IRP) logs, collected from Continella *et al.* [1]. The authors performed a large-scale IRP data collection containing 1.7 billion IRPs produced by $2,245$ different applications. In their study, they developed an IRPLogger (or I/O file system sniffer) to collect this low-level file system logs. The data collection was executed in two phases: (1) Benign IRP logs collection - the researchers managed 11 volunteers that included developers, home, and office users. The volunteers installed the built sniffer tool on their machines, ranging from Microsoft Windows 7 to Windows 10, in order to record the daily activities. These recorded logs were considered ground truth dataset based on the assumption that none of these machines were compromised with any kind of cyber-attacks; and (2) Ransomware IRP logs collection - the authors used 383 ransomware samples and collected IRP logs for each ransomware execution inside a Windows 7 machine. It is to note that not every process among these execution traces was malicious, and hence, we had to extract ransomware processes from each ransomware dataset separately. In summary, the basis of our analysis is built upon the IRP data collection performed in [1] for both 11 voluntary users (benign records) and 272 ransomware samples collected over approx. 90 minutes of executions. We were not able to use all of 383 ransomware samples because of unexpected formatting errors encountered while processing the raw IRP logs, such as, timestamps, string to integer conversion, etc. We will revisit the dataset processing issues in future.

[2]https://docs.microsoft.com/en-us/windows-hardware/drivers/ifs/irps-are-different-from-fast-i-o

III. Empirical Study

A. Data Cleaning and Formatting

Once we acquire the dataset, we start with data cleaning and formatting. The IRP logs came in a tab separated document. We discuss its default features categorized in groups for better understanding, along with the ones that we derive -

Types of IRP Operation. As mentioned in Section II(A), there are three types of IRP operations: IRP, FIO, and FSF. In the dataset, we omit records that do not relate to any one these mentioned IRP operations, such as, err. We then perform One-Hot Encoding for these three types of categorical operations, *i.e.*, transform the operation strings into one-hot numeric array.

Operation Time. Each log in the dataset comes with two timestamps: pre-operation time and post-operation time. We convert the text formatted timestamps (H:M:S:f) into milliseconds and then compute the time difference between them by introducing an additional feature, called 'operation time elapsed.'

IDs. Each operation is associated with several numerical IDs, such as, parent id, process id, and thread id. It is worth mentioning that the values of each field is only unique to each process as long as the process is active. Upon completion of the process, the IDs are freed by the Operating System (OS), and the OS can reassign the same IDs to another process. The combination of process id and process name allows to isolate the activities of a particular process over the span of a ransomware session or in a ransomware IRP dataset.

Flags. As mentioned in section II(A), each IRP operation is featured with different types of flag values. All the flag values are in hexadecimal format. The flags include IRP Flag, Major Operation Type, Minor Operation Type, Transaction, Status, Inform, and Argument 1 - Argument 6. Each IRP Flag contains a hexadecimal value to indicate its value, and four other types of flag values with a space, such as, No Cache, Paging I/O, Synchronous API, and Synchronous Paging I/O. An example

TABLE I
DATA DISTRIBUTION OF NOTABLE FEATURE SPACES FOR OF VICTIM MACHINE'S IRP LOGS DURING RANSOMWARE EXECUTION (APPROX. 90 MINUTES).

Ransomware Family (Sample Size)	Class	Types of IRP Operations			Flags			File System Information				
		IRP Operation	FSF Operation	FIO Operation	IRP Flag	Unique Major Operation Type	Status	File Object	Unique File Accessed	Total File Accessed	Buffer Length	Entropy
Critroni	Malicious	119,028	191	257	16	**17**	13	5,467	10,251	303,082	988	3,965
(2)	Benign	190	66	63	8	11	4	40	21	354	529	11
Crowti	Malicious	219,734	116,364	1,166	16	15	11	11,225	16,660	224,861	17,621	23,042
(23)	Benign	266	162	290	9	11	4	48	22	830	556	66
CryptoDefense	Malicious	131,344	101,837	867	16	15	11	11,999	17,264	212,562	**22,765**	24,329
(6)	Benign	268	136	190	9	11	4	48	21	826	599	61
CryptoWall	Malicious	131,444	95,052	1,015	16	15	11	8,079	15,994	228,009	5,708	16,884
(17)	Benign	263	146	192	8	11	4	47	22	804	526	60
Dalexis	Malicious	217,532	200	980	14	15	10	8,905	17,431	427,810	1,147	2,912
(1)	Benign	227	118	207	8	11	4	43	21	811	459	70
Deshacop	Malicious	500,284	136,120	7,843	14	15	12	5,158	19,766	508,256	3,162	21,810
(2)	Benign	256	144	196	9	11	4	47	21	787	472	65
High	Malicious	417,447	1,178	544	13	15	10	11,716	19,712	**711,972**	19,381	44,084
(1)	Benign	221	116	152	8	11	4	42	21	662	370	40
Parite	Malicious	406,462	101,702	6,321	14	15	12	5,246	16,921	412,897	3,384	19,457
(1)	Benign	256	148	389	9	11	4	46	21	813	499	65
Processhijack	Malicious	291,645	84,697	4,359	14	15	12	5,806	13,449	296,118	2,141	13,989
(1)	Benign	238	94	139	8	11	4	40	21	582	478	34
Pwszbot	Malicious	96,803	85,176	490	15	15	11	9,626	15,473	262,780	5,617	15,825
(1)	Benign	256	116	127	8	11	4	49	22	577	512	22
Seven	Malicious	175,124	152,269	942	16	15	11	10,637	17,509	272,084	15,649	23,448
(1)	Benign	264	132	157	8	11	4	46	22	805	502	56
TeslaCrypt	Malicious	611,678	**267,387**	26,953	14	15	11	**32,708**	30,564	638,757	3,759	21,472
(1)	Benign	250	94	86	8	11	4	42	21	519	579	30
Tinba	Malicious	**653,943**	194,433	**27,630**	14	15	**14**	9,044	**35,162**	681,707	12,042	**47,949**
(1)	Benign	255	165	420	8	11	4	46	21	812	466	74
Tpyn	Malicious	177,600	65,748	1,612	16	16	11	7,546	10,917	179,402	6,206	16,158
(1)	Benign	260	135	154	8	11	4	48	22	707	448	22
Upatre	Malicious	131,771	99,159	905	16	15	11	10,650	16,267	202,118	10,417	19,381
(56)	Benign	265	147	195	8	11	4	48	21	826	527	62
Vobfus	Malicious	297,730	138	444	12	14	7	11,280	14,193	438,984	6,722	20,453
(1)	Benign	220	84	137	7	11	4	32	21	427	350	30
Yakes	Malicious	177,218	92,796	1,159	16	15	11	8,869	13,101	192,897	8,105	19,401
(150)	Benign	266	148	198	9	11	4	48	22	827	535	61
Zbot	Malicious	166,258	104,951	888	16	15	11	10,629	16,457	208,110	13,302	22,757
(6)	Benign	266	135	163	8	11	4	48	22	780	559	53
Median	Malicious	197,566	97,106	998	15	15	11	9,335	16,558	284,101	6,464	19,955
Results	Benign	256	135	176	8	11	4	47	21	796	507	58

value of this feature is $0x00000404 - -\text{S}-$, which signifies the value of the IRP Flag itself is $0x00000404$, No Cache is 0 (for 1 the first character after space would be N), Paging I/O is 0 (for 1 the second character would be P), Synchronous API is 1, and Synchronous Paging I/O is 0 (it would be Y for 1). Thus, we generate four additional features from one. We ensure that all the string-based hexadecimal values are properly formatted, *i.e.*, removed extraneous white spaces. We also handle missing values in the flags' features.

File System Information. In regards to process interaction with the file system, we notice several important features, such as, File Object, Device Object, File Name, Entropy, and Buffer Length. Similar to flags, both File Object and Device object are in hexadecimal format. The datatype for Entropy is in float while Buffer Length is in integer.

B. Ransomware Family Labeling

One of the important tasks related to ransomware detection is to identify the family name of all the ransomware samples as we aim to design our experimentation on the basis of ransomware family (described in Section IV and V). All the 272 ransomware samples were represented uniquely by their SHA256 hashes. We utilize VirusTotal API Engine [13] to scan and identify the samples' hashes and received a complete scan report containing results aggregated from many Anti-Virus (AV) tools, such as, Kaspersky, Symantec, etc. If the scanned sample was detected malicious, the AV engines labeled it with a malware family name, *i.e.*, Kaspersky generated the label *Yakes* for the ransomware samples having "00ce22ce923e246990e43289b8b5b8191cbfc28dbee6d30b66226df0aa14b7bd" SHA-256 hash. We observe, the labels provided by different AV engines[3] were not same for the same signature hashes. Therefore, we explore a generalized approach to assign one family label to samples from the same family as scanned by different AV engines. Hence, we make use of AVClass - a malware labelling tool [10]. We feed the scan report for all the samples in a JSON file to the AVClass tool. Then, the tool assigns one (most probable) family name to a set of similar samples as diagnosed by VirusTotal. We thereby obtain 18 ransomware families for all the samples used in our research.

C. Data Processing

After performing data cleaning and formatting as well as mapping 272 ransomware samples to 18 ransomware families, we locate the *actual* ransomware IRP logs from the ransomware dataset. As mentioned earlier, not every process is a malicious process captured in the ransomware session. Thus, we apply a heuristic approach. We start by grouping the logs with process id and process name as this gives us a complete trace of an application's activities in the session

[3]https://www.virustotal.com/gui/file/00ce22ce923e246990e43289b8b5b8191cbfc28dbee6d30b66226df0aa14b7bd/detection

TABLE II
DATA DISTRIBUTION OF NOTABLE FEATURE SPACES FOR BENIGN USERS' IRP LOGS DURING A RANDOM SESSION (10 MINUTES).

User Type	Operating System Version	Types of IRP Operations			Flags			File System Information				
		IRP Operation	FSF Operation	FIO Operation	IRP Flag	Unique Major Operation Type	Status	File Object	Unique File Accessed	Total File Accessed	Buffer Length	Entropy
Developer	Windows 10	129	10	36	6	9	3	17	11	166	537	2
Home	Windows 8.1	281	36	68	7	10	4	51	21	605	46	3
Office	Windows 10	143	18	87	6	8	4	18	9	412	610	13
Home	Windows 7	51	0	20	6	6	2	11	4	78	141	1
Home	Windows 7	190	48	26	7	10	4	28	26	314	919	**84**
Developer	Windows 10	92	24	40	6	9	4	27	18	169	1,233	5
Developer	Windows 8.1	276	4	78	6	7	2	18	25	549	3,380	33
Home	Windows 8.1	**414**	38	83	9	12	**7**	37	36	**765**	**7,474**	54
Home	Windows 8.1	354	**100**	**212**	**13**	12	6	**68**	**47**	683	4,459	36
Home	Windows 7	161	4	14	5	8	4	9	5	203	798	25
Office	Windows 7	170	13	27	5	8	4	10	14	205	2,270	17
Median Results		170	18	40	6	9	4	18	18	314	919	25

until the process is terminated. During ransomware execution inside a test environment, Continella *et al.* [1] ensured no other process(es) would access the files in certain folders in the file system, *i.e.*, they plotted decoy files in the machine to keep track of this. Based on this assumption, we compute the number of total and unique files accessed by each process in the ransomware dataset. In addition, we flag the process if the process accesses the decoy files in the system. With use of the flag and the two features' values, it becomes fairly easier for us to identify the ransomware processes. Then, we extract all ransomware process id(s) and process name(s) from each sample's IRP logs. Then, we introduce a 'class' feature in the dataset to label benign and malicious instances. Thus, we craft a supervised setting for the ransomware IRP logs dataset.

D. Observations

In this section, we discuss some of the key insights gained while analyzing benign and ransomware dataset.

Ransomware IRP Logs. We notice, the ransomware dataset reflects approx. 90 minutes of IRP logs. We present Table I to show the data distribution of the notable feature spaces in all 18 different ransomware family datasets containing both malicious and benign logs. We take the median values for each feature per family, *i.e.*, *Crowti* ransomware family has 23 ransomware samples, and we compute the median results from all its samples. To better portray our results and findings, we partition our observations in three categories: Types of IRP Operations, Flags, and File System Information. We present the median results from all of the listed feature values at the bottom of the Table I. The difference in every feature enables us distinctly separate malicious and benign logs for every case. We compile this tabular data distribution format for the entire execution of the ransomware; however, it will be promising to see the change of the feature values over time, *e.g.*, Time Interval Analysis, to analyze the trend of the feature space over time. We highlight the largest value of each feature to conveniently identify which ransomware family it represents.

Benign IRP Logs. We further compare the data distribution space of ransomware with the benign IRP logs (as shown in Table II). We randomly pick a 10-minute uninterrupted session from each user type machine log in order to explore a benign user's interaction with his/her file system. File System Information features gives us notable insights from the table between benign users' logs and ransomware logs, *i.e.*, the number of unique files and total files accessed by the benign user is significantly low than any ransomware process in our study. It strengthens our intuition that benign user profiling would enable ransomware defenders and researchers to isolate the anomalous processes at first. If the anomalous process accesses decoy file(s) at the same timeframe, then it will be an indication the process is malicious ransomware in our case.

IV. EXPERIMENTAL METHODOLOGY

Artificial Neural Network (ANN). Once the dataset is processed and labeled with a binary class (benign or malicious), we leverage Artificial Neural Network (ANN or commonly known as Neural Networks) to learn the underlying patterns of both benign and ransomware impacted IRP logs and build a detection model for ransomware detection with the capabilities of generalization and transfer learning, *i.e.*, we aim to employ an ANN structure that can effectively adapt itself for unseen data and perform prediction on related context(s) based upon learning from a given context. Therefore, our goals are set to efficiently perform detection for (1) all the ransomware families (trained and tested over all the samples); (2) one ransomware family by being trained over all the samples of a different family; and (3) each family (trained and tested over a single family in every iteration). To achieve these goals, we construct the ANN with one input layer, one hidden layer, and one output layer. Our built ANN is a fully connected network, where the size of the input layer neuron is the number of features in the training set. We evaluate the performance of the model with three different hidden layer settings: (1) between the size of the input layer and the size of the output layer; (2) $2/3$ of the size of the input layer, plus the size of the output layer; and (3) less than twice the size of the input layer. We observe the third tuning setting is more suitable for our purpose as we achieve better empirical findings. The output layer contains one neuron for the binary classification task providing class prediction probability.

Experimental Setup. To further describe ANN's configuration in our experimentation, we utilize Rectified Linear

Unit (ReLU) activation function, where the function and its derivative are monotonic with a range of 0 to ∞, for both input and hidden layer. We use Sigmoid activation function, where the S-shaped function is differentiable with a range of 0 and 1, for the output layer. We leverage an Adam Optimization Algorithm and Binary Cross Entropy (BCE) loss function for model compilation. We incorporate the early stopping method during training to ensure the generalization ability of the network. We monitor validation loss for up to 3 iterations to trigger this action if the model shows no learning development in the training phase. We select 20% of the training records as the validation set to perform this task.

Before we train the model, further dataset processing needed to be done. From Flags-based features, we drop all the Arguments and Inform features as they do not help predict the class. We additionally remove Process Name and File Name features as we aim to build an ANN model that will not rely on such name variables. We perform One-Hot Encode to the remaining hexadecimal decimal based features, *e.g.*, IRP Flag, IRP Major Operation Type, IRP Minor Operation Type, Status, and Transaction and store the final combined dataset. As the data distribution of the feature space is not normalized, we perform Min-Max Scalar operation on the dataset. We split the dataset into training (80%) and testing (20%) set in a stratified fashion, *i.e.*, maintaining the equal distribution of binary classes. We combine both benign and ransomware IRP logs in different settings (described in the Results section) to evaluate our built ANN model's detection capabilities.

We then train the model with the derived training set. we select 128 batch size for faster execution and 100 epochs for the training process. It is to note that we do not find the model to be trained over the entire 100 epochs due to the incorporation of the early stopping method, which ensures that the model is neither underfitted nor overfitted. After the model is trained, we evaluate its performance with the derived testing set. Along with the accuracy of the model, we compute the precision score, recall score, and F_1 score to give us a better understanding of the model's performance in every setting we design. In the spirit of open science, our implementation is open source for the community with MIT License on GitHub[4].

V. Results

In this section, we discuss the following experimental findings from executing the Artificial Neural Network (ANN) model in the following iterations maintaining the same architecture and hyperparameter settings of the ANN -

Single Family-wise Iteration. We design this experimentation primarily in two ways: (1) when the number of the ransomware samples of a particular ransomware family is equal or greater than five (*e.g.*, CryptoDefense, Upatre, etc.), we train the ANN model with 80% of the samples while we test it with the remaining 20%; and (2) when the number of ransomware samples of a particular ransomware family is less than five (*e.g.*, Critroni, Deshacop, etc.), we combine all the samples' logs together and then perform 80%-20% train-test split. The rationale behind this is ensuring randomness as much as possible for each family. The same approach is followed for all of the eighteen families and the results are presented in Table III. To summarize the model's performance, the median scores of the families' results are reported, which shows that the model performs with 99.86% accuracy, 99.84% precision score, 99.87% recall score, and 99.84% F_1 score. *This demonstrates, the built ANN model can effectively detect variants of ransomware samples for each ransomware family.*

TABLE III
Performance of the binary classification using ANN for single ransomware family wise iteration.

Ransomware Family	Sample Size	Accuracy	Precision Score	Recall Score	F_1 Score
Critroni	2	0.9987	0.9976	0.998	0.9978
Crowti	23	0.996	0.9978	0.9965	0.9972
CryptoDefense	6	0.9983	0.9981	0.9988	0.9985
CryptoWall	17	0.9988	0.9989	0.9988	0.9984
Dalexis	1	0.9987	0.9988	0.9988	0.9983
Deshacop	2	0.9988	0.9988	0.9989	0.9989
High	1	0.9988	0.9973	0.9986	0.9984
Parite	1	0.9986	0.9986	0.9987	0.9986
Processhijack	1	0.9987	0.9987	0.9988	0.9988
Pwszbot	1	0.9988	0.9981	0.9985	0.9983
Seven	1	0.9988	0.9988	0.9988	0.9988
TeslaCrypt	1	0.9984	0.9988	0.9953	0.9975
Tinba	1	0.9986	0.9982	0.9989	0.9985
Tpyn	1	0.9985	0.9981	0.9987	0.9989
Upatre	56	0.9989	0.9988	0.9987	0.9982
Vobfus	1	0.9984	0.9986	0.9985	0.9981
Yakes	150	0.9975	0.9967	0.9985	0.9981
Zbot	6	0.9983	0.9981	0.9988	0.9984
Summary	272 (total)	0.9986 (median)	0.9984 (median)	0.9987 (median)	0.9984 (median)

Training with One Family, Testing with Another. We continue the experimentation with a different setting by training the ANN model with randomly chosen four samples from one ransomware family, *e.g.*, Upatre and Yakes, and testing it with a randomly chosen sample from another ransomware family, *e.g.*, Crowti and Zbot, respectively. Table IV shows the results for these two cases. There is no rationale behind selecting these two particular sets of ransomware families. Further investigation shows that the results remain similar in almost every case, with performance scores of the model within the range of $99.7\% \pm 0.2\%$. *This indicates, the constructed ANN model can effectively detect the underlying pattern of a new ransomware family on which it was not trained over.*

TABLE IV
Performance of the binary classification using ANN for training with one family, testing with another.

Training Family	Testing Family	Accuracy	Precision Score	Recall Score	F_1 Score
Upatre	Crowti	0.9976	0.9969	0.9987	0.9983
Yakes	Zbot	0.9989	0.9984	0.9988	0.9981

Training and Testing with All the Families. The final experiment includes all the ransomware families; however, inclusion of 18 ransomware families with all 272 ransomware samples turned out to be infeasible in terms of memory usage. Therefore, one randomly selected sample from each of the ransomware families were used in the experiment. We train the model with 80% of the combined dataset while test it

[4]www.github.com/TnTech-CEROC/irp-logs-mining

with the remaining 20%. We achieve 99.8% accuracy, 99.74% precision score, 99.86% recall score, and 99.85% F_1 score. *This signifies, our constructed ANN model can effectively detect variants of ransomware families in our study.*

VI. Related Work

Ransomware researchers have utilized this low-level I/O Request Packet (IRP) logs to propose various defensive mechanisms to combat against ransomware. Kharraz *et al.* [6] analyzed $1,359$ ransomware samples to describe workings and effect of ransomware, and then, monitored file system through IRP logs in users' machine for successful ransomware detection. Since then, many state-of-the-art research work in this field incorporated IRP to their study for ransomware protection where IRP logs were used as a tool for file system and / or process monitoring [1], [5], [9], [11].

McIntosh *et al.* [7], [8] conducted ransomware behavior analysis using the same dataset [1] like us. The authors in both research work extracted some useful pieces of information from the dataset: (1) types of the files (as well as paths) ransomware generally targeted; (2) number of the IRP requests to modify file contents; and (3) the total number of file types modified during ransomware operation. To compare, our work has covered much more granular level insights on ransomware behavior, along with construction of an efficient Artificial Neural Network (ANN) model for ransomware detection.

VII. Conclusion

Detection of ransomware, a special purpose malware suite, has been one of the prime research areas among malware analysis researchers as well as malware defenders due to the severe damage it incurs to the real-world organizations. As new variants of ransomware surface in the wild quite frequently, different areas are explored to tailor efficient detection schemes, among which I/O Request Packet (IRP) logs is considered as one of the most promising tools. In this research, we utilize IRP logs from 272 ransomware samples (belonged to 18 ransomware families) and 11 benign machines used in [1]. Having the logs processed, we devise a data-driven approach for effective ransomware detection by learning at granular level with actionable insights of ransomware behavior. We incorporate data analytic tasks to preview data distribution of all the studied ransomware samples. We construct Artificial Neural Network (ANN or Neural Networks) architecture as the detection scheme. In order to evaluate the model, we design three experimentation settings, and our results show that the ANN model can effectively (1) detect variants of ransomware samples for each ransomware family; (2) discover the underlying pattern of a new ransomware family on which it was not trained over; and (3) predict IRP logs from variants of ransomware families. We report the accuracy, precision score, recall score, and F_1 score for every experiment we have performed, and the computed results fall in the range of $99.7\% \pm 0.2\%$ for each experimental setting.

Our approaches are not suitable for real-time detection since it trains on post-ransomware infection IRP logs; however, we plan to devise an early detection scheme by testing the built model's performance within 5, 10, or 20 minutes of the ransomware activities. We will explore finding threshold-based values on different feature spaces as well as common sequences in the ransomware's IRP flag-based features that could lead to an effective early detection. With the acquisition of several ransomware families' logs, we consider performing multiclass classification in our future work as well.

Acknowledgement

The work reported in this paper has been entirely supported by Cybersecurity Education, Research & Outreach Center (CEROC) at Tennessee Tech University.

References

[1] Andrea Continella, Alessandro Guagnelli, Giovanni Zingaro, Giulio De Pasquale, Alessandro Barenghi, Stefano Zanero, and Federico Maggi. Shieldfs: a self-healing, ransomware-aware filesystem. In *Proceedings of the 32nd Annual Conference on Computer Security Applications*, pages 336–347, 2016.

[2] Microsoft Docs. Example i/o request - an overview - windows drivers, June 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/example-i-o-request---an-overview.

[3] Ivan Firdausi, Alva Erwin, Anto Satriyo Nugroho, et al. Analysis of machine learning techniques used in behavior-based malware detection. In *2010 Second international conference on advances in computing, control, and telecommunication technologies*, pages 201–203. IEEE, 2010.

[4] Hyun Cheol Jeong, Chae Tae Im, and Joo Hyung Oh. Malware auto-analysis system and method using kernel callback mechanism, March 29 2012. US Patent App. 12/942,700.

[5] Amin Kharaz, Sajjad Arshad, Collin Mulliner, William Robertson, and Engin Kirda. {UNVEIL}: A large-scale, automated approach to detecting ransomware. In *25th {USENIX} Security Symposium ({USENIX} Security 16)*, pages 757–772, 2016.

[6] Amin Kharraz, William Robertson, Davide Balzarotti, Leyla Bilge, and Engin Kirda. Cutting the gordian knot: A look under the hood of ransomware attacks. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 3–24. Springer, 2015.

[7] Timothy McIntosh, Julian Jang-Jaccard, Paul Watters, and Teo Susnjak. The inadequacy of entropy-based ransomware detection. In *International Conference on Neural Information Processing*, pages 181–189. Springer, 2019.

[8] Timothy R McIntosh, Julian Jang-Jaccard, and Paul A Watters. Large scale behavioral analysis of ransomware attacks. In *International Conference on Neural Information Processing*, pages 217–229. Springer, 2018.

[9] Shagufta Mehnaz, Anand Mudgerikar, and Elisa Bertino. Rwguard: A real-time detection system against cryptographic ransomware. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 114–136. Springer, 2018.

[10] Marcos Sebastián, Richard Rivera, Platon Kotzias, and Juan Caballero. Avclass: A tool for massive malware labeling. In *International Symposium on Research in Attacks, Intrusions, and Defenses*, pages 230–253. Springer, 2016.

[11] Daniele Sgandurra, Luis Muñoz-González, Rabih Mohsen, and Emil C Lupu. Automated dynamic analysis of ransomware: Benefits, limitations and use for detection. *arXiv preprint arXiv:1609.03020*, 2016.

[12] Nathan Shone, Tran Nguyen Ngoc, Vu Dinh Phai, and Qi Shi. A deep learning approach to network intrusion detection. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2(1):41–50, 2018.

[13] VirusTotal. Public api v2.0, 2019. URL: https://www.virustotal.com/en/documentation/public-api/.

[14] Christina Warrender, Stephanie Forrest, and Barak Pearlmutter. Detecting intrusions using system calls: Alternative data models. In *Proceedings of the 1999 IEEE symposium on security and privacy (Cat. No. 99CB36344)*, pages 133–145. IEEE, 1999.

[15] Adam Young and Moti Yung. Cryptovirology: Extortion-based security threats and countermeasures. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 129–140. IEEE, 1996.