Malware Family Classification based on Behavioral Profiles and Key System Calls

Tsung-Ying Chen SID: 861310198, tchen063@ucr.edu

Contribution to Project	Project: 50% 1.1~1.2: (1)(3) 2.5, 2.6, 2.7 3 4.1, 4.3 5.1, 5.3 6.1, 6.3

Ting Hsun Lin SID: 861310545, tlin035@ucr.edu

Contribution to Project	Project: 30% 1.1~1.2: (2) 2.1 3 4.2 5.2 6.2

Chen Han Ho

Contribution to Project	Project: 20% In process tree parts, and modified some detail in tools and others.

ABSTRACT:

Some malwares grouping researches [3] take advantage of well-labeled malware families and focus on proposing better methods to classify malwares. However, in real scenario, we often face unknown malwares or some variants of malwares. One purpose of the project is towards clustering unknown malwares by profiling their system calls numbers. In addition, we want to find the key system calls that can be used to classify malware family.

1. INTRODUCTION:

1.1 Problem Definition:

Group same behaviors into a malware family becomes a critical issue in malware detection and defense. One purpose of this paper is to propose a convincing clustering structure based on profiling the runtime behavior of malwares in terms of system calls numbers. Then, the Silhouette analysis [4] and sum of square are proposed to evaluate the goodness of different malware family clustering results.

The second purpose is training a more accurate classifier for malware family classification based on system calls behavior. In addition, we want to extract the key system calls that can be used for malware family classification effectively.

Our study provides a solution includes router behavior profiling, behavior clustering and clustering evaluation and key system call selection. Which are different from most of the previous works in malware detection or behavior analysis.

To be more specific, there are four main issue that this paper deal with.

(1) Validate the observations from previous paper [1]: if infected routers exhibit an initial spike and an overall 50% increase in the number of system calls.
(2) Clustering distinct groups of malwares based on profiling the runtime behavior of malwares in terms of system calls numbers. This analysis could suggest the convincing cluster number of malwares.
(3) Malware family Classification by nearest neighbor based on system calls. In addition, implement the feature selection algorithm—forward search to find the key system calls that classify the malware family most effectively.
(4) Building the process tree for profiling by the degree of the process. Let's in different architecture, when some unknown malware come into the specific architecture, could we identify to some family?

The input to the problem is:

- System calls log data of infected routers, provided by RARE [2].

The desired output is:

(1) Figures and tables, based on the system calls log, that can validate the observations from previous paper
(2) Suggest the convincing clustering of malware family structure. Try to differentiate their behavior based on our observation and previous researches.
(3) A key system calls set that classifies malware family effectively. The result should accompany with great accuracy.
(4) In each malware family, we test in three different architectures, to create a unique process tree.

1.2 Contributions:

We propose to develop a method to Analyze router behavior by profiling the system calls numbers. Specifically, our contributions are expected to be:

(1) Show that if infected routers exhibit an initial spike and an overall 50% increase in the number of system calls by proposing the profiling work. And find some new information.
(2) Propose the convincing cluster numbers among malwares of BE (MIPS, Big-Endian) based on their system calls profiling.
(3) Propose the key features for classifying malware family with the corresponding accuracy.
(4) Propose the process tree to identify the unknown malware in different architecture.

The rest of this paper is structured as follows. In section 2, we review some previous works relevant to malware analysis, definition of behaviors, K-Means clustering, Feature Selection algorithm and Nearest Neighbor. In section 3, we introduce the raw data produced by RARE which is the input of this research. In section 4, we demonstrate all the analysis results. In section 5, discuss the results. In section 6, give the conclusions.

2. BACKGROUND:

2.1 Malware Analysis:

Traditional malware analysis for detection and classification falls into two general types: static and dynamic analysis. In static analysis, strings of bytes associated with malware samples are discovered through reverse engineering and used as a signature for identifying malicious software. However, there are two drawbacks of static analysis. First, static analysis is often prone to high false positive rates due to evolution in code basis and code repacking. Second, it is required for reverse engineering to generate reliable and meaningful signatures.

On the other hand, dynamic and behavior based analysis aims to provide methods for effectively and efficiently extracting unique patterns of each malware family based on its behavior. Malware samples of the same family often use the same code base, provide the same functionality using the same behavioral events [5], and so on. In dynamic analysis, behavior patterns can be represented by various symbols, such as API call and system call. Moreover, researchers previously proposed various detection and classification methods for malware analysis based on their behavior, including API call-based and system call-based methods. API call-based detection methods cannot generate distinct signatures until decompilation or disassembly process is completed, which is often expensive. System call based detection methods can more accurately detect malicious behavior than other methods, since it is impossible to modify original functionality of system calls: malware creators always attempt to disguise malicious behavior as normal behavior.

2.2 Definition of Behaviors of Malware:

Christodorescu et al. [6] view the running program as a black-box and specify program behavior as its interaction with the operating system. In other words, a unique behavioral pattern is the set of operating system calls that this process invokes. Every action that involves communication with the process' environment (e.g., accessing the file system, sending a packet over the network, or launching another program) requires the process to take advantage of an appropriate operating system service.

2.3 RARE:

Darki, A. *et al.* [2] proposes, RARE (Riverside's Augmented Router Emulator), a systematic approach to analyze the behaviors of router malwares. The RARE fools malware binaries to activate irrespective of their preferred target platform. From a practical point of view, RARE has the ability to run the malware on an emulated router and consists of the following modules, whose goal is to: a) perform static analysis on the malware to extract information for its execution, b) instantiate an emulated router with hints on what configuration the malware wants to see, c) inject malware into the router and fool it to activate, d) replay arbitrary network traffic or in response to malware requests, and e) monitor the malware and provide information to subsequent runs of the same malware. If the malware fails to activate, we repeat the process, and the last step provides information that guides the new execution. This paper use 221 anonymous router-specific malwares binaries.

2.4 Clustering Methods and Evaluation:

In this paper, we classify the dataset using K-Means clustering algorithm. K-Means clustering algorithm is a data mining technique that can be applied to group the dataset into k clusters [8].

Let $X = \{x_1, x_2, x_3, \dots, x_n\}$ be the set of data points and $C = \{c_1, c_2, c_3, \dots, c_n\}$ be the set of centers. Then, randomly select k cluster centers. Next, calculate the distance between each data point and cluster centers. After that, specify the data to the cluster center whose distance from the cluster is the minimum of all cluster centers. Subsequently, recalculate the new cluster center using formula

$$c_i = (\frac{1}{n_i})\sum_{j=1}^{n_i} x_j$$

where, n_i represents the number of data points in the ith cluster. Finally, recalculate the distance between each point and obtained new cluster centers. Stop if no point was reassigned; otherwise repeat again the process of specifying the data point to the cluster center whose distance from the cluster is the minimum of all cluster centers.

The Silhouette analysis [7] can be used to study the separation distance between the resulting clusters. The silhouette plot displays a measure of how close each point in one cluster is to points in the neighboring clusters and thus provides a way to evaluate parameters like number of clusters visually.

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a_i) and the mean nearest-cluster distance (b_i) for each sample. The Silhouette Coefficient for a sample is

$$S_i = (\frac{b_i - a_i}{\max\{a_i, b_i\}}).$$

To clarify, b is the distance between a sample and the nearest cluster that the sample is not a part of. This measure has a range of [-1, 1].

The Silhouette coefficients near +1 indicate that the sample is far away from the neighboring clusters. A value of 0 indicates that the sample is on or very close to the boundary between two neighboring clusters and negative values indicate that those samples might have been assigned to the wrong cluster.

2.5 Behavior of System Calls of Infected Router

From the previous research conducted by Darki, A. *et al.* [2], it demonstrates that infected routers exhibit: (a) an initial spike and (b) an overall 50% increase in the number of system calls. Since we try to validation these two observation, the behavior of system calls of infected router, in this paper, is the frequency of the system calls.

2.6 Forward Feature Selection Algorithm

Previous researches provided a method that builds dependence graphs as the behaviors of malware [6]. We can use these dependence graphs to detect malware [10] or classify malware by matching their behavior graph [11]. However, we try to use an easier way to classify malware family --nearest neighbor classifier with appropriate features selected by forward search algorithm. [13]

The forward feature selection procedure begins by evaluating all feature subsets which consist of only one input attribute. In other words, we start by measuring the Leave-One-Out Cross Validation (LOOCV) error of the one-component subsets, $\{X_1\}$, $\{X_2\}$, ..., $\{X_M\}$, where M is the input dimensionality; so that we can find the best individual feature, $X_{(1)}$. Next, forward selection finds the best subset consisting of two components, $X_{(1)}$ and one other feature from the remaining M-1 input attributes. Hence, there are a total of M-1 pairs. Let's assume $X_{(2)}$ is the other attribute in the best pair besides $X_{(1)}$. Afterwards, the input subsets with three, four, and more features are evaluated. According to forward selection, the best subset with m features is the m-tuple consisting of $X_{(1)}$, $X_{(2)}$, ..., $X_{(m)}$, while overall the best feature set is the winner out of all the M steps.

In this paper, we view different malware family as different classes and each kinds of system calls are our feature.

2.7 Nearest Neighbor

Within the forward selection algorithm, we need a classifier to predict the class of an unknown test sample. In this paper, we choose Nearest Neighbor classifier with Euclidian Distance to classify malware family.

1-NN is the most intuitive nearest neighbor type classifier. The one nearest neighbor classifier that assigns a point x to the class of its closest neighbor in the feature space.

3. DATA AND TOOLS:

The raw dataset is reproduced from RARE [2]. There are 3 architectures of system, ARM, MIPS(BE) and MIPS(LE). The number of malicious router emulation results are ARM (42), MIPS (63) and MIPS (30).

3.1 Data Preprocessing

In this paper, all preprocessing and analyzing of K-Means Clustering take advantage of scikit-learn package [7]. The raw input of each malware emulation result is a series of consecutive system calls sequence. First, transform this raw data into one data point with multiple features and each of them is the specific system call count that appeared in raw data. Subsequently, merge all malware emulation results into one metric for each architecture (ARM, BE and LE). In this metric each row means one malware emulation result and each column indicate one specific system call. All values in metric means the number of system call counts.

And the preprocessing of data for router behavior profiling and Feature Selection are done by Tsung-Ying Chen with matlab. As Table.1 showed, the input data is simply the number of each kind of system call over 1300ns, and the ground truth class number, provided by Chuang, C.Y., is manually compared over the control flows of 53 disassembled MIPS big-endian malware.

	class	fork	write	open	…
aesDDoS-mips-MIPS-AES-DDos.ELF.MIPS.mmd	1	6126	6697	5275	…
aesDDoS-mips-MIPS-AES-ddwrts.AESddos.MIPS.mmd	2	10467	7726	6631	…

Table. 1: Part of input dataset: 53 big-endian malware samples with their true class and system calls frequency

4. ANALYSIS AND RESULTS:

4.1 Previous observations validation

(a) The number of system calls of an infected router is not always higher

We found 21 out of 66 BE-architecture samples have less that 30% increase in the number of system calls compared to the benign router. And as Table 2 showed, there 5 samples of infected routers have similar number of system calls as benign router. And 3 samples are obviously lower.

Infected router sample	# system calls over 1300ns
Benign	91,353
aesDDoS-mips-MIPS-AES-fff85.AESddos.MIPS.mmd	63,525
bash0day-D_161003-ntpd	92,937
bash0day-Fgt150923-slnth	92,641
bash0day-fgt_150908-output8.1	94,809
bash0day-IoT_161010-IoTmips	92,884
dofloo-Linux.Dofloo-mips-MIPS-DOFLOO-Linux.Dofloo.2	74,117
elknot-mips-MIPS-und-ARM-MIPS-ELKNOT-awqs.MIPS.ElknotCrypt.mmd	70,678
kluh-mips-MIPS-KLUH-xiao	93,298

Table 2: number of system calls

(b) Not always initial spike on infected router

We found 6 out of 42 ARM-architecture samples and 7 out of 66 BE-architecture samples have no initial spike. Fig. 1 is a non-initial-spike sample.

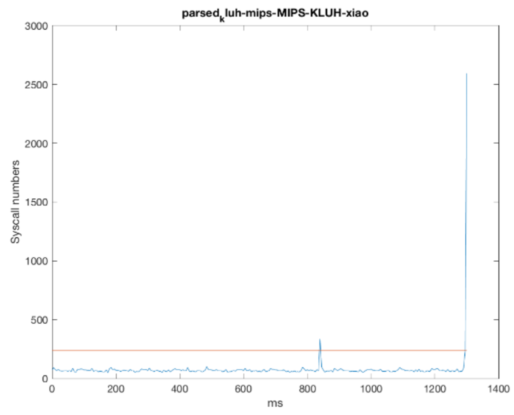


Fig. 1: no-initial-spike infected router

In addition, we found that even the benign router has an initial spike. As Fig. 2 showed, it's hard to tell the difference between a benign and an infected router based on their spike or even their system calls frequency.

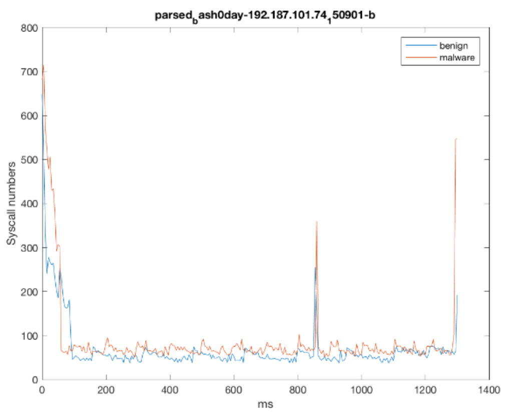


Fig. 2: Behavior of benign and infected routers

(c) Special system calls behavior

On Table 2, there are three infected router samples that have significant less system calls. And these three infected router samples have very different behavior as Fig. 3 showed.

1. Before the middle spike, they perform low frequency of system calls
2. Longer middle spike

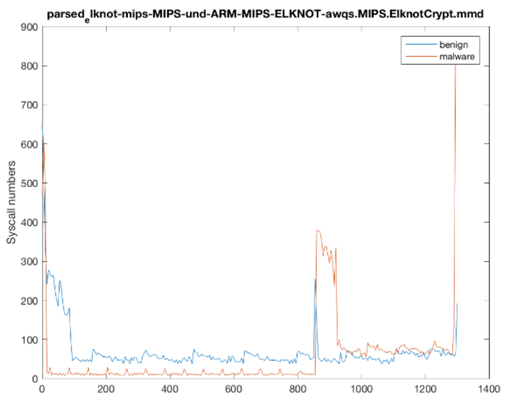


Fig.3: Special behavior of specific infected router

4.2 K-Means Clustering Analysis

We take advantage of Silhouette Analysis [7] to determine the number of clusters on K-Means clustering. As the Fig.4. shows. The x-axis is increasing numbers of clusters from 2 to 12 and the y-axis are corresponding silhouette coefficient score. The peak of silhouette coefficient score is .70 when cluster number equal to 9. The silhouette coefficients range between [-1, +1]. This coefficient closer to +1 indicate that the sample is more far away from the neighboring clusters relative to its own cluster. Therefore, according to the Fig, the result of silhouette analysis suggests 9 clusters among malwares of BE.

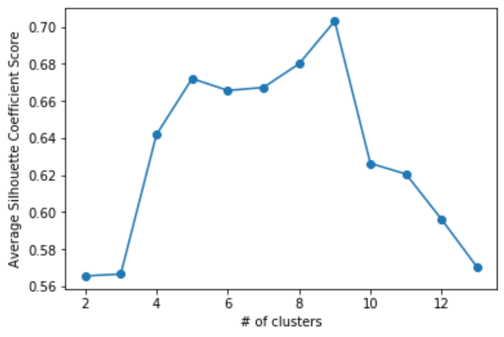


Fig.4. Silhouette coefficients for increasing number of clusters

Also, in Fig.5., we can observe that the total sum of square decrease with the increasing number of cluster.

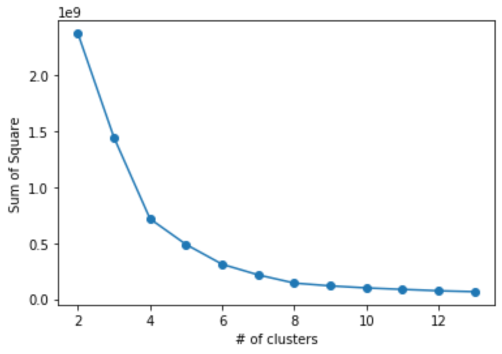


Fig.5. Total sum of square for increasing number of clusters

The Fig.6 is the result of 9 clusters silhouette analysis. The x-axis indicates the silhouette coefficient score and the y-axis are distinct clusters. The Fig demonstrate corresponding distributions of clusters. The vertical red dotted line indicates the average silhouette score (.70) among 9 clusters. The Fig shows that most cluster (cluster 0, 1, 2, 3, 8) are above the average silhouette score (.70). In other words, most clusters are far away from the neighbor cluster than average level.

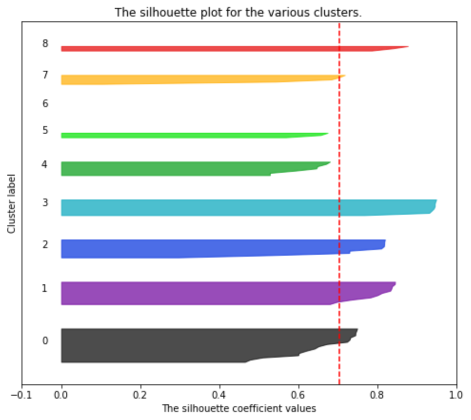


Fig.6. 9 clusters silhouette analysis

4.3 Key system calls selection

we use feature selection algorithm—forward search with nearest neighbor to select the key features for predicting the malware family and then use leave-one-out validation to calculate the accuracy.

The leave one out validation gets the accuracy: 94.3% with 7 kinds of system calls (features). This result is better than the clustering method [12]. Table 3 shows the feature set (system call number) and their corresponding accuracy.

Feature set (system call)	accuracy
{ nanosleep}	84.9%
{ nanosleep, _llseek }	90.5%
{ nanosleep, _llseek, fcntl }	92.5%
{ nanosleep, _llseek, fcntl, time, lseek, rename, read, gettimeofday }	94.3%

Table 3: selected feature set with accuracy

We also use backward elimination and exhaustive search to select key system calls for classification. As table 4 showed, the selected features and the corresponding accuracy are similar.

	Best features	Accuracy
Forward	{nanosleep, _llseek}	94.30%
Backward	{clone, _llseek}	90.50%
Exhaustive (within 3 features)	{nanosleep, _llseek, fcntl}	92.50%

Table 4: features selected by three different features selection algorithms

4.4 Process tree analysis

In the experiment, the input file is the malicious program. Then try to trace each of the system call that have a connection between each pid. That is mean if one processor create a child by fork() or clone(), I create a tree node with the first node that the produce at the beginning of the program. And connect each the node create by their parents. Further, to record the number of each level.

Malware	Level	Total
AES-1	1,1,18,21	41
AES-49	1,6,7,3	17
AES-arm	1,6,7,3	17
AES-azc	1,6,7,3	17
AES-fff83	1,6,7,3	17
AES-linux	1,6,7,3	17
AES-Srarm	1,6,8,3	18
AES-sysdate	1,10,0,0	11
AES-unpacked1	1,1,17,22	41
AES-wangs	1,1,17,22	41
Bash0day	1,2,0,0	3

Table5: Results for the algorithm that trace all the process to get each number of the level tree. And the total number for each malware in the ARM architecture.

First, the AESDDOs is the test malware in Table.5. And we can observe that come with the number of each tree level would be mostly 1,6,7,3. However, there is some exception, like AES-unpacked1 and AES-wangs would be put in same characteristic, Otherwise, would be classify as same characteristic. Although AES-1 performed lots of node at level2 and level3, most of the node would irrelevant to the behavior of the program (by going through the whole system call).

Malware	Level	Total
AES-MIPS	1,1,10,0	12
AES-lywrt	1,1,10,0	12
AES-ELF	1,1,10,0	12

Moose-mips	1,1,4,0	6
Mrblack-mips	1,1,10,0	12
Bash0day	1,2,0,0	3

Table6: Results from the BE architecture.

Malware	**Level**	**Total**
AES-MIPS	1,6,7,3	17
AES-lywrt	1,6,7,3	17
AES-ELF	1,6,7,3	17
Bash0day	1,2,0,0	3

Table7: Results from the LE architecture

And now compare BE & LE in Table6 & Table7. We can easily see that in the same malware but in different structure, they will get different number of each level.

Although in same structure, but with LE and BE, shows the determination in different address may impact how the process generate.

5. DISCUSSION:

5.1 Previous observations validation

First, the statement "an overall 50% increase in the number of system calls" may not be a specification for infected routers. In fact, the number of system calls of infected routers fluctuate a lot. As table 8 showed, it could be -30%~90% increase compared to the benign router.

Infected router sample	system calls increased compared to benign router
A	26.3%
B	-30.5%
C	41.7%
D	84.5%
E	1.7%
F	-22.6%
…	…

Table 8

Second, we found that there are about 13% of infected router didn't cause an initial spike. Instead, the benign router shows an initial spike. Therefore, initial spike may not be a specification of infected router.

Third, we found that some infected routers perform low frequency of system calls before middle spike and have a long middle spike. These two behaviors are not found in other router samples. So they may have some relationships.

5.2 K-Means Clustering Analysis

The Fig.7. shows behavioral profile of 9 clusters. Each line indicates one malware emulation result. As you can see, they all portrait similar system calls profile. Therefore, 9 clusters are a convincing family grouping structure of BE. Moreover, considering the expert validation, Darki, A also suggest 8 or 9 clusters in BE based on his own observations.

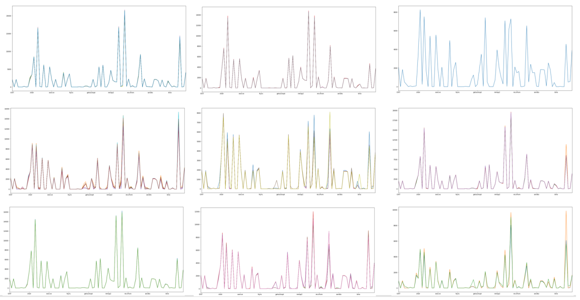
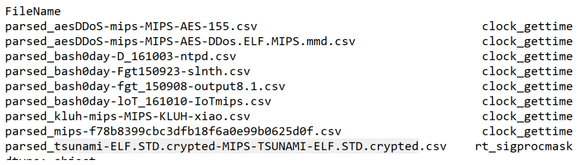


Fig.7. Behavioral Profile of 9 clusters

For each cluster, we checked the most frequent system call. *read(), wait4(), open() and clock_gettime()* appeared most frequently among all clusters. However, there is one special system call, *rt_sigprocmask()*, has highest frequencies in *"tsunami-ELF.STD.crypted-MIPS-TSUNAMI-ELF.STD.crypted"*. *rt_sigprocmask()* is used to fetch and/or change the signal mask of the calling thread. The signal mask is the set of signals whose delivery is currently blocked for the caller. Therefore, keep calling *rt_sigprocmask()* could force the router keep deliver (unblock signal) packages. This might be the one specific behavioral pattern for the "*tsunami*".



```
FileName
parsed_aesDDoS-mips-MIPS-AES-155.csv                                       clock_gettime
parsed_aesDDoS-mips-MIPS-AES-DDos.ELF.MIPS.mmd.csv                         clock_gettime
parsed_bash0day-D_161003-ntpd.csv                                          clock_gettime
parsed_bash0day-Fgt150923-slnth.csv                                        clock_gettime
parsed_bash0day-fgt_150908-output8.1.csv                                   clock_gettime
parsed_bash0day-loT_161010-IoTmips.csv                                     clock_gettime
parsed_kluh-mips-MIPS-KLUH-xiao.csv                                        clock_gettime
parsed_mips-f78b8399cbc3dfb18f6a0e99b0625d0f.csv                           clock_gettime
parsed_tsunami-ELF.STD.crypted-MIPS-TSUNAMI-ELF.STD.crypted.csv          rt_sigprocmask
```

Fig.7. The most frequent system calls of one cluster

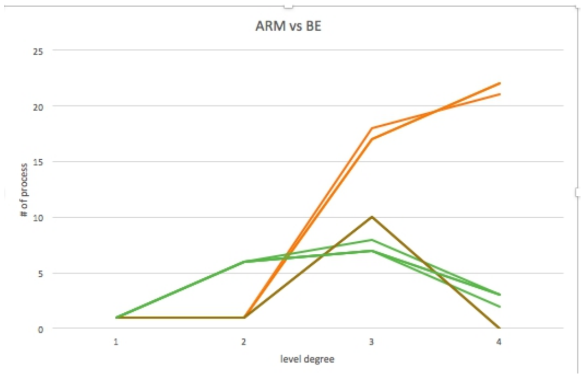
5.3 Key system calls selection

Based on the same data, there is already an unsupervised method that using clustering to classify malware family [12]. Nonetheless, this feature selection algorithm finds the key system calls for classification and therefore performs a better accuracy.

As Table 3 showed, we can achieve 85% accuracy when only use nanosleep to classify malware family. In fact, no other system call gets more than 80% and most system calls are less than 50%. Therefore, we can conclude that different malware families may have different frequency on system call—nanosleep. In addition, previous work [12] also chose nanosleep as one of the key features. Another key system call is _llseek. It has been selected by many feature selection algorithms.

5.4 Process Tree Analysis

In our experiment, in the AES-DDos can easily to separate the ARM & the BE architecture, but for the Bash0day cannot identify its different. Further, we tried to build the dependent graph for each malware, however, it doesn't work well in our environment, so we have a gap for the explanation of why is the same malware in different come with same tree.



6. CONCLUSION:

6.1 Previous observations validation

According to our result, the two observations from previous work may not be an effective way to identify infected router. So we try to propose another method for classification.

In addition, the characteristics that low frequency of system calls before middle spike with a long middle spike could be a feature to identify infected malware. And we may find some relationship about why it happened.

6.2 K-Means Clustering Analysis

This paper applied the Silhouette analysis combine with sum of square to decide the number of clusters for unkown-labled malware. Furthermore, the analysis result suggests 9 clusters in BE.

Also, we find one specific behavioral pattern for the "*tsunami*" which keeps calling "*rt_sigprocmask()*" for forcing the router keeping delivering.

There are other classification methods based on the ordering of system calls, such as dependency graph and n-gram. However, the implementations are more complicated. In the future, compare both methods, system calls number based and system calls ordering based. If both methods have closer performance in classifications of malwares, the clustering based on the number of system calls count is preferred because of fast implementation.

6.3 Key system calls selection

Based on the feature selection algorithm, we proposed 7 key system calls, showed on table 3, that can be used to classify the family of malware. The accuracy of leave-one-out cross validation is 94.3%. In addition, we propose a key system call—nanosleep that different malware family may have different behavior on it.

6.4 Process Tree

Although process tree seems well in some case, but if we want to get more accurate, we need to do more effort of the tree. There been some gap that explain the process tree to identify different architecture, it seems well in some case, however, there can not be the most directly impact form the process tree. Further, we try to build a whole graph of each system call.

7. REFERENCES:

[1] Darki, A., Duff, A., Qian, Z., Naik, G., Mancoridis, S., & Faloutsos, M. Don't Trust Your Router: Detecting Compromised Routers. Proceedings of the 2016 ACM

[2] Ahmad Darki, Chun-Yu Chuang, Michalis Faloutsos, Zhiyun Qian, Heng Yin, RARE: A Systematic Augmented Router Emulation for Malware Analysis. (to appear) Passive and Active Measurement(PAM) 2018

[3] Hsiao, S. W., Sun, Y. S., & Chen, M. C. (2017). Virtual Machine Introspection Based Malware Behavior Profiling and Family Grouping. arXiv preprint arXiv:1705.01697.

[4] Peter J. Rousseeuw. "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics, 1987, 20: 53–65.

[5] Mohaisen, A., West, A. G., Mankin, A., and Alrawi, O. Chatter: Classifying malware families using system event ordering. In IEEE Conference on Communications and Network Security, CNS 2014, San Francisco, CA, USA, October 29-31, 2014, pp. 283-291.

[6] M. Christodorescu et al., "Mining Specifications of Malicious Behavior," in Proc. India Software Engineering Conference (ISEC), 2008, pp. 5–14.

[7] Silhouette Analysis on KMeans Clustering, http://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html

[8] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. 2005. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[9] Michael Dymshits, Benjamin Myara, David Tolpin. Process Monitoring on Sequences of System Call Count Vectors

[10] Clemens Kolbitsch., et al.: Effective and Efficient Malware Detection at the End Host. USENIX security symposium (2009), 351-366.

[11] Younghee Park., et al.: Fast Malware Classification by Automated Behavioral Graph Matching. Cyber Security and Information Intelligence Research (2010)

[12] Chun-Yu Chuang., et al.: Router Malware Clustering (in processing). (2018)

[13] Kan Deng: OMEGA: On-Line Memory-Based General Purpose System Classifier. (1999) Retrieved from https://www.cs.cmu.edu/~kdeng/research.html