INVESTIGATION OF AWSCTD DATASET APPLICABILITY FOR MALWARE TYPE CLASSIFICATION

Doc. Nikolaj Goranin PhD.[1], Dainius Čeponis[1]
Vilnius Gediminas Technical University, Saulėtekio al. 11, 10223 Vilnius, Lithuania
dainius.ceponis@vgtu.lt

Abstract: *Nowadays, information systems security is a crucial aspect – vulnerable system endpoint can lead to severe data loss. Intrusion detection systems (IDS) are used to detect such unfortunate events. Implementation place defines the type of IDS: network-based (NIDS) for network traffic monitoring or host-based (HIDS), to detect malicious actions on the host level. IDS can be effective only if generated alerts are correctly evaluated and classified, what is typically done by a trained staff, but requires a lot of time and human resources. While a lot research is done with NIDS alerts evaluation, HIDS research is lacking behind. HIDS reported operating system calls could be used to define the importance of alarms and steer analysts to the most critical issues. In this article we demonstrate the applicability of our created Attack-Caused Windows System Calls Traces Dataset (AWSCTD), which is currently the most comprehensive dataset of system calls generated by almost all modern malware types, for training different classification methods on malware type recognition and later alert prioritization. The effectiveness of different classification methods is evaluated, and results are presented. Currently achieved results allow to decrease the load on analytical staff, dealing with malware classification and related alert prioritization by 92.4%, which makes this approach applicable for practical use.*

Keywords: HIDS, ALERT PRIORITISATION, MACHINE LEARNING, WINDOWS, SYSTEM CALLS

1. Introduction

With the significant growth of computer usage in daily tasks – computer systems security question raises naturally. It is important to detect the intrusion and react immediately to minimize the consequences of an attack. One of the first publication, describing the need of intrusion detection system as a component to protect infrastructure, was a United States Air Force report published in 1972 by James Anderson [1]. The report stated that IDS system can be used to minimize the amount of manual work that has to be done for log and audit data analysis – which is a time and personnel consuming task. Later, in 1985, the first real-time IDS based on expert-written rules, was presented [2]. There are two main types of IDS systems: network-based intrusion systems (NIDS) and host-based intrusion systems (HIDS). NIDS are deployed on the network level, but are not able to protect the company if intrusion was able to overcome it and reached the victim system by the means of malware, exploit, etc. HIDS is deployed on the host level and deals with the threats defined above.

Todays IDSes are more sophisticated than just use of a collection of well-crafted rules. Often, they use various machine learning (ML) methods to detect unauthorized entry or malicious activity. They incorporate three main detection methods [3]: anomaly-based, signature-based and hybrid. The anomaly-based method is good against Zero-day attacks but has a high false-positive rate. The signature-based method correctly detects known attacks, but new intrusions are often missed. The hybrid method combines the best practices of the other two. Manual security personnel work is still needed to adequately detect and classify and prioritize an intrusion. Each alert comes with a significant amount of data to be analyzed. The research about artificial intelligence (AI) and IDS merge conducted in 1994 stated that “a user typically generates between 3 - 35 Megabytes of data in an eight hour period and it can take several hours to analyze a single hour’s worth of data” [4]. That amount of data much larger in the current version of IDS implementations: mainly because more information is gathered and increased traffic generated by a user.

In general, after IDS deployment, security analyst has not just to deal with the number of alerts but also with an extensive amount of data produced by those alerts. All that information has to be reviewed and analyzed making that task time-consuming and error-prone. Alerts count minimization and most critical attacks prioritization are the main approaches of the IDS management [5]. However, alerts amount minimization task is insufficiently analyzed for HIDS, and all the latest research and effort are focused on NIDS [5]–[8]. For that reason, in this paper we suggest using application generated system calls sequences as one of the components to classify the malware detected by HIDS that can be later used for alert importance prioritization. Alert that came from an application which has 92% similarity to Trojan must be reviewed before WebToolbar generated alert because of a different risk level caused by these malware type. Various ML methods, applied to the AWSCTD [9], which is currently the most comprehensive dataset of system calls generated by almost all modern malware types, were generating commercially acceptable accuracy level of more than 80%.

This paper is divided into two main sections. First, related work and information about IDS alert management are introduced. The second part will present AWSCTD usage for malware classification in the means of Windows OS system calls sequences: how the training is executed, what results is achieved and explanation of them. Finally, the conclusions are provided.

2. Related Work

A system call is an API used by the computer program to request a service from the kernel of the operating system: for opening a file, running a thread, writing to the registry or opening a network connection. Because of that origin (a primary artifact of the OS kernel), system call data is a very popular choice for malware research and detection.

The most simple and the cheapest by the meaning of computing power is the frequency-based features classification for intrusion detection. One of the latest approaches is the ADFA-WD dataset analysis. Authors of the dataset achieved the 72% intrusion detection rate on the preliminary analysis of dataset when Naïve Bayes Net were used as classifiers [10].

System call sequences are another popular method to represent features, which is costly but generates strong detection metrics [11]. The main idea is that system call is a “word” and sequences made of them form meaningful phrases. Since tasks of different application differ, that information can be used to classify them correctly (for example if it is a malware or legitimate application). Hidden Markov Models (HMM), Recurrent Neural Networks (RNN) and Convolutional Neural Networks (RNN) are the top ML methods used for this task type.

Two types of operations have been introduced, to solve the problem of IDS generated alerts classification [5]:

- *Low-level* (also called prioritization) operations deal with each alert individually to enrich its attributes or assign a score to it based on potential risk.

• *High-level* (also called aggregation) alert management techniques, such as aggregation, clustering, correlation, and fusion, were proposed to deal with sets of alerts and provide their abstraction.

Both techniques have some disadvantages. High-level suffers from including alerts that are not significant. That leads to inappropriate results and increased load. The main problem with low-level techniques is that they have to automatically examine all the generated alerts and prioritize them for further inspection, which also leads to improperly classified items [12].

In general, alert prioritization operations use information from various domains that can be gathered from the network. Security policy, network topology, vulnerability analysis of network services and installed software, as well as asset profiles are the mostly used factors affecting the prioritization of alerts [13]–[16]. The proposed systems use internal databases, which provide all required information about monitored network topology. The security analyst can manually manage databases or the system itself can automatically update them with the help of Nmap tool [17].

Some attempts were made to automate malware classification by system calls [18]. However, authors do not motivate that task from the point of view of further attack risk prioritization, and the size of malware dataset they used was twice smaller than in case of our experiments.

To enrich alert prioritization for the HIDS, we propose to use a system call sequences of executed applications as one of the determinants. System calls can be used to determine if the executed application belongs to one of the malware types, such as trojans, worms, bots, and ransomware, etc..

3. AWSCTD usage for the malware classification

AWSCTD was generated with the help of publically available malware collection from VirussShare.com site. Since malware can use code obfuscation and packers in order to hide from static analysis, we have used the dynamic malware analysis to gather data about malware behavior. Custom made system was specially implemented to collect system calls sequences on Windows 7 operating system [9].

The total of 5 malware families was selected from the dataset for our experiment. The main selection criteria – family should have more than 100 samples of unique family representatives and corresponding system calls sequences. Kaspersky provided family descriptor was used. Table 1. shows the number of unique malware samples in each family ("DangerousObject" category according to the Kaspersky: Malicious software is detected by KL Cloud Technologies. This verdict is used for samples that were not classified exactly.).

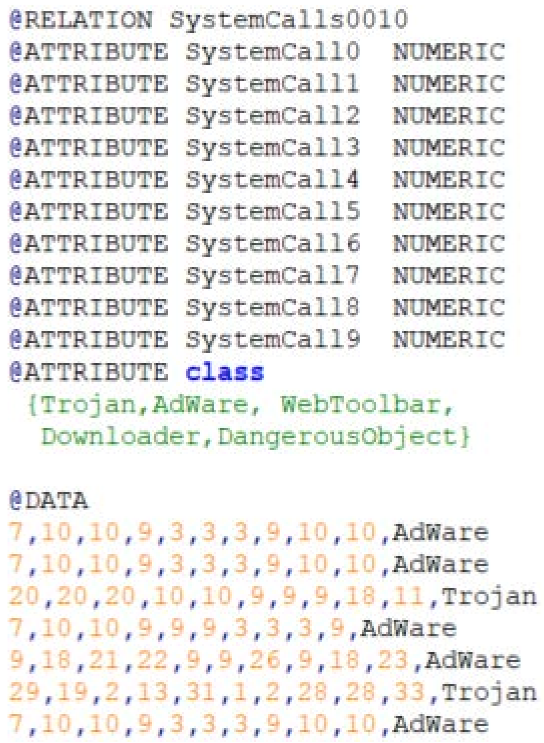
Table 1. *The number of each malware label.*

Label	Count
Trojan	1755
AdWare	4333
WebToolbar	618
Downloader	710
DangerousObject	105
Total:	7521

The different size of families represented helps us to evaluate the influence of the size of initial dataset on the accuracy of classification.

3.1 Feature preprocessing and selected ML methods

We are stating that it is enough to test the first batch off application system calls to determine its family. For that reason, the first 10, 20, 40, 60, 100, 200, and 400 system calls were used to generate the training and testing sets. The sample for 10 first system calls can be seen on Fig. 1. Every system call is represented by the unique number and sequence of these numbers forms the feature vector used for training.



```
@RELATION SystemCalls0010
@ATTRIBUTE SystemCall0  NUMERIC
@ATTRIBUTE SystemCall1  NUMERIC
@ATTRIBUTE SystemCall2  NUMERIC
@ATTRIBUTE SystemCall3  NUMERIC
@ATTRIBUTE SystemCall4  NUMERIC
@ATTRIBUTE SystemCall5  NUMERIC
@ATTRIBUTE SystemCall6  NUMERIC
@ATTRIBUTE SystemCall7  NUMERIC
@ATTRIBUTE SystemCall8  NUMERIC
@ATTRIBUTE SystemCall9  NUMERIC
@ATTRIBUTE class
 {Trojan,AdWare, WebToolbar,
  Downloader,DangerousObject}

@DATA
7,10,10,9,3,3,3,9,10,10,AdWare
7,10,10,9,3,3,3,9,10,10,AdWare
20,20,20,10,10,9,9,9,18,11,Trojan
7,10,10,9,9,9,3,3,3,9,AdWare
9,18,21,22,9,9,26,9,18,23,AdWare
29,19,2,13,31,1,2,28,28,33,Trojan
7,10,10,9,3,3,3,9,10,10,AdWare
```

Fig. 1. *WEKA data file fragment for the first ten system calls.*

A WEKA 3.8 data mining software was used to test various classification methods [19]. None of the WEKA provided default method values were changed or tinkered to achieve better results. Different type of Weka provided ML methods were selected to test the accuracy of malware type classification. Methods with the training values can be seen in Table 2.

Table 2. *Used ML methods with training parameters.*

Method	Training parameters
Bayes Net	-D -Q bayes.net.search.local.K2 -- -P 1 -S BAYES -E bayes.net.estimate.SimpleEstimator -- -A 0.5
Naïve Bayes Net	
Support Vector Machines	-S 0 -K 2 -D 3 -G 0.0 -R 0.0 -N 0.5 -M 40.0 -C 1.0 -E 0.001 -P 0.1 -model \"C:\Program Files\Weka-3-8\" -seed 1
Neural Networks	-L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
Decision Table	-X 1 -S \"BestFirst -D 1 -N 5\"
J48 (C4.5)	C 0.25 -M 2
LMT (Logistic Model Tree)	-I -1 -M 15 -W 0.0

3.2 Results and discussion

Classifiers were trained and tested with a 10-fold cross-validation technique. Cross-validation is a technique to evaluate predictive models by partitioning the original sample into a training set to train the model, and a test set to evaluate it. Our selected out of the box ML methods allowed to achieve results suitable for practical application. Almost all of them have achieved more than 80% off accuracy for the dataset with the best result of 92.4 by SVM. The percentage of correctly classified malware samples by different methods can be seen in Table 3.

Table 3. *Correctly classified percentage of instances.*

Feat. count	Bayes Net	Naïve Bayes	SVM	NN	Dec. Table	J48	LMT
10	**87.5**	61.6	89.4	87.2	88.1	88.5	89.1
20	73.8	60.9	89.4	87.8	88.0	88.8	89.2
40	74.3	61.7	91.6	88.5	89.4	90.7	91.0
60	75.0	66.1	91.8	90.2	89.3	90.9	91.2
100	75.4	61.8	**92.4**	**91.4**	90.0	**92.1**	**92.1**
200	76.1	68.2	89.8	90.3	90.0	91.9	91.8
400	75.9	**69.3**	87.5	84.8	89.7	92.1	92.1

The best results of classification are represented by the green color in the Table 3. The worst values are in red.

Worst results were obtained by using Bayes classification family: Bayes Net and Naïve Bayes Net. Naïve classification method produces the worst of all results: achieving only 69.8% of correct classification. SVM has achieved the best result of 92.4% with the 100 system call long feature vector. It is clear that 100 features vector is the most optimal subset to achieve more than 90% accuracy. Decision trees algorithms have also produced remarkable results for feature vector length of 100 and 400: it was even better than NN and had an accuracy of 92.1%. That is expected since already in 2008 the C4.5 decision tree algorithm was one of the best ten choices for the data mining [20].

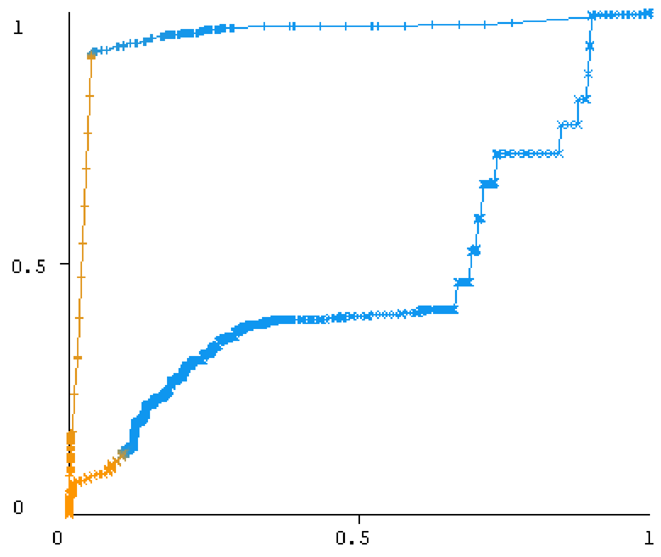
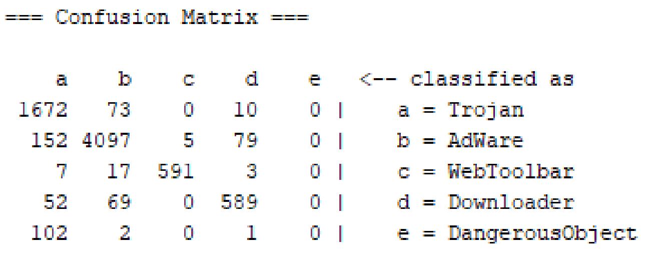


Fig. 2. *ROC curve of J48 (+) and Naive Bayes (x) classifiers. X-axis - False Positive Rate. Y-axis - True Positive Rate.*

Classification performance of J48 vs. Naïve Bayes can be seen on Fig.2. J48 area under ROC (AUC) (0.9486) is bigger than Naïve Bayes (0.4664). AUC of Naïve Bayes indicates that it produces worse results than random classification [21] which produces AUC of 0.5.

Obviously, the best classification results were obtained for malware families that had the biggest number of the samples. In this specific dataset that were Trojans and AdWare (1755 and 4333 samples respectively). Family “Dangerous object”, a family with only 105 samples of in fact different types, has shown the lowest detection rate. SVM Confusion Matrix for the 100 features vector can be seen in Fig 3.



```
=== Confusion Matrix ===

    a    b    c    d    e   <-- classified as
 1672   73    0   10    0 |    a = Trojan
  152 4097    5   79    0 |    b = AdWare
    7   17  591    3    0 |    c = WebToolbar
   52   69    0  589    0 |    d = Downloader
  102    2    0    1    0 |    e = DangerousObject
```

Fig. 3. *SVM confusion matrix of the best result.*

When comparing training and testing execution times, decision trees have shown the best results. Both training and testing values are among the shortest. SVM and NN show the worst outcome, even though they produce only 1.3% and 0.3% better performance compared to J48 regarding accuracy. On the contrary NN is 24 times faster than SVM since they are trained, and testing stage is activated. Weka produced elapsed time for training and testing can be seen in Table 3 and Table 4.

Table 4. *Weka produced Elapsed_Time_training values.*

Feat. count	Bayes Net	Naïve Bayes	SVM	NN	Dec. Table	J48	LMT
10	0.05	0.01	1,3	8,4	0,4	0,1	14,2
20	0.07	0.01	1,9	17,2	0,9	0,2	21,1
40	0.09	0.02	3,8	46,3	2,8	0,4	40,2
60	0.29	0.04	4,8	90,4	4,2	0,8	69,2
100	0.31	0.06	7,5	222,4	5,5	1,4	106,6
200	0.93	0.12	16,8	938,3	16,0	3,2	301,0
400	3.61	0.28	40,6	4410,1	33,7	9,1	726,0

Table 5. *Weka produced Elapsed_Time_testing values.*

Feat. count	Bayes Net	Naïve Bayes	SVM	NN	Dec. Table	J48	LMT
10	0,003	0,009	0,130	0,002	0,002	0,005	0,003
20	0,002	0,017	0,184	0,002	0,003	0,000	0,002
40	0,003	0,031	0,309	0,009	0,000	0,002	0,003
60	0,011	0,041	0,396	0,009	0,002	0,003	0,000
100	0,019	0,078	0,586	0,024	0,000	0,000	0,003
200	0,045	0,154	1,303	0,094	0,002	0,000	0,006
400	0,100	0,454	3,098	0,413	0,005	0,000	0,016

The smallest amount of time is represented with the green color and the worst results for training and testing – with the red color.

All tests were performed on a system with an Intel i5-4670 3.40 GHz CPU. In comparison with the [18] used TITAN X GPU, it is a much slower device in the means of computation power. Also, we were using ML methods without any parameters optimization and much smaller features vectors for the training and testing, and, all system calls were used for the feature vector generation (authors of [18] have not included repetitive system calls from their dataset) which provide more information for the ML methods. All these differences let us achieve results that can be also used practically. That leads to the conclusion, that Deep Learning methods are an overhead for a task of this type – correctly classified instanced are over 90% and training time is more suited for the domain that generates new samples every day (model proposed by [18] has a training time that ranges from three to ten hours).

4. Conclusions

A theoretical approach to use malware generated systems calls sequences for IDS reported alerts prioritization was discussed in this paper. The new, Windows OS based malware generated system calls dataset AWSCTD was used to show the applicability of this approach. Different types of ML methods were applied to classify malware by family. The results have showed that this approach could be used in practical applications – the best detection rate of 92.4 was obtained by using SVM with the 100 features vector. Comparable results were achieved by the use of decision tree algorithms: J48 and LMT. They generated 92.1% classification results. J48 has also showed the best training and testing times, that is very useful when new samples must be introduced for training. The obtained results of time needed for model training flicker, that deep learning or NN techniques are in fact superfluous for that specific task. Generated results suggest, that 100 first malware requested system calls is more than enough to achieve more than 90% classification accuracy, that are typically more than in enough for practical use. ML methods were used without any tinkering – with WEKA proposed default values. This implies for the future work, i.e. tuning configuration parameters for most perspective methods.

5. References

[1] J. P. Anderson, "Computer Security Technology Planning Study," October, vol. 2, no. 93, 1972.

[2] D. E. Denning and P. G. Neumann, "Requirements and model for IDES-a real-time intrusion detection expert system," Document A005, SRI International, vol. 333. 1985.

[3] A. L. Buczak and E. Guven, "A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection," IEEE Commun. Surv. TUTORIALS, vol. 18, no. 2, 2016.

[4] J. Frank, "Articial Intelligence and Intrusion Detection : Current and Future Directions," Proc. 17th Natl. Comput. Secur. Conf., vol. 10, pp. 1–12, 1994.

[5] K. Alsubhi, E. Al-Shaer, and R. Boutaba, "Alert prioritization in Intrusion Detection Systems," NOMS 2008 - IEEE/IFIP Netw. Oper. Manag. Symp. Pervasive Manag. Ubiquitous Networks Serv., pp. 33–40, 2008.

[6] T. H. Nguyen, J. Luo, and H. W. Njogu, "An efficient approach to reduce alerts generated by multiple IDS products," Int. J. Netw. Manag., 2014.

[7] R. Shittu, A. Healing, R. Ghanea-Hercock, R. Bloomfield, and M. Rajarajan, "Intrusion alert prioritisation and attack detection using post-correlation analysis," Comput. Secur., vol. 50, pp. 1–15, 2015.

[8] C. M. Chen, D. J. Guan, Y. Z. Huang, and Y. H. Ou, "Anomaly network intrusion detection using Hidden Markov Model," Int. J. Innov. Comput. Inf. Control, vol. 12, no. 2, pp. 569–580, 2016.

[9] D. Čeponis and N. Goranin, "Towards a Robust Method of Dataset Generation of Malicious Activity for Anomaly-Based HIDS Training and Presentation of AWSCTD Dataset," Balt. J. Mod. Comput., vol. 6, no. 3, 2018.

[10] W. Haider, G. Creech, Y. Xie, and J. Hu, "Windows based data sets for evaluation of robustness of Host based Intrusion Detection Systems (IDS) to zero-day and stealth attacks," Futur. Internet, vol. 8, no. 3, 2016.

[11] T. R. Glass-Vanderlan, M. D. Iannacone, M. S. Vincent, Qian, Chen, and R. A. Bridges, "A Survey of Intrusion Detection Systems Leveraging Host Data," pp. 1–40, 2018.

[12] F. Valeur, G. Vigna, C. Kruegel, and R. A. Kemmerer, "A comprehensive approach to intrusion detection alert correlation," IEEE Trans. Dependable Secur. Comput., vol. 1, no. 3, pp. 146–168, 2004.

[13] R. Sadoddin and A. Ghorbani, "Alert correlation survey," in Proceedings of the 2006 International Conference on Privacy, Security and Trust Bridge the Gap Between PST Technologies and Business Services - PST '06, 2006, p. 1.

[14] B. Morin, L. Mé, H. Debar, and M. Ducassé, "M2D2: A formal data model for IDS alert correlation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2516, pp. 115–137, 2002.

[15] P. A. Porras, M. W. Fong, and A. Valdes, "A mission-impact-based approach to INFOSEC alarm correlation," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 2516, pp. 95–114, 2002.

[16] E. Chakir, M. Moughit, and Y. I. Khamlichi, "Building an Efficient Alert Management Model for Intrusion Detection Systems," vol. 3, no. 1, pp. 18–24, 2018.

[17] G. Lyon, "Nmap: the network mapper," Nmap.org, 2018. .

[18] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 9992 LNAI, pp. 137–149, 2016.

[19] I. H. Witten, E. Frank, and M. A. Hall, Data Mining: Practical machine learning tools and techniques. 2005.

[20] X. Wu et al., "Top 10 algorithms in data mining," Knowl. Inf. Syst., vol. 14, no. 1, pp. 1–37, 2008.

[21] T. Fawcett, "An introduction to ROC analysis," Irbm, vol. 35, no. 6, pp. 299–309, 2005.