Detecting Scanning Computer Worms Using Machine Learning and Darkspace Network Traffic

Nelson Ochieng
Faculty of Information Technology
Strathmore University
Nairobi, Kenya
nochieng@strathmore.edu

Waweru Mwangi
Faculty of Information Technology
JKUAT
Nairobi, Kenya
wmwangi@icsit.jkuat.ac.ke

Ismail Ateya
Faculty of Information Technology
Strathmore University
Nairobi, Kenya
iateya@strathmore.edu

Joseph Orero
Faculty of Information Technology
Strathmore University
Nairobi, Kenya
jorero@strathmore.edu

***Abstract*—The subject of this paper is computer worm detection in a network. Computers worms have been defined as a process that can cause a possibly evolved copy of it to execute on a remote computer. They do not require human intervention to propagate; neither do they need to attach themselves to existing files. Computer worms spread very rapidly and modern worm authors obfuscate their code to make it difficult to detect them. This paper proposes to use machine learning to detect them. The paper deviates from existing approaches in that it uses the darkspace network traffic attributed to an actual worm attack to validate the algorithms. In addition, it attempts to understand the threat model, the feature set and the detection algorithms to explain the best combination of features and why the best algorithms succeeds where others have failed.**

Keywords—computer worm detection, malware detection, machine learning, darkspace network traffic, behavioral computer worm detection

I. INTRODUCTION

Malicious code includes computer virus, Trojan horse, spy-ware, ad-ware, computer worms among others. This present research limits itself to computer worms and specifically computer worm detection in a network. Reference [1] defines a computer worm as "a process that can cause a (possibly evolved) copy of it to execute on a remote computational machine."

Several different computer worm detection approaches have been explored in the research environment. Some of these works involve static analysis of malicious code where malware is analyzed without executing it while others involve dynamic analysis where the behavior of malware is analyzed as it interacts with the system. Also, some of the approaches are content payload based while others are behavior based.

Among the problems with the existing approaches are the high false positive and high false negatives. Reference [2] explains that this could partly be because of the existing approaches relying on only one parameter for detection.

Automated detection approaches are to be encouraged since computer worm spread is always rapid as explained in [3]. In addition, modern day worms are especially difficult to detect because worm authors employ complex mutations to evade detection and use code obfuscation techniques such as polymorphism and metamorphism [4 & 5].

Reference [6] explains that most research efforts in using machine learning for computer worm detection are directed towards removing the redundancy and noise from the data collected, performing efficient training for the classifier by using real variants of worms and identifying the most optimum classifier among the data mining classification algorithms. It uses anomalies with DNS requests and responses as the discriminating feature.

We argue that high detection accuracy and confidence can be achieved by better characterization of computer worms using multi parameters. In addition, machine learning classifiers learning from past empirical evidence of computer worm attacks can be used for future prediction and classification of unseen instances.

The dataset used for machine learning training and testing in this work is attributed to actual worm attack and therefore suitable to deliver useful research validity.

This work attempts to follow the advice presented in [7].

It makes the following contributions:

First, the research explains the threat model, the network environment where the detection is to happen and the implication of false negatives.

Secondly, the research investigates the relevance of the feature set used for the detection and explains their

significance. The features are ranked to explain their contribution to the detection.

Third, the research investigates the classification capability of different machine learning algorithms on the dataset. It attempts to explain why various algorithms succeed in some cases and why they fail in some cases. The research identifies the best performing algorithm on this dataset and tries to explain why it outperforms the others based on the feature set.

The rest of the paper is organized as follows. Section II reviews existing literature on computer worm detection using machine learning. Section III discusses the methodology for the research starting with a review of the dataset used and the machine learning algorithms used. It then explores the dataset and the features in that dataset and discusses the experiments together with the tuning parameters. Section IV then discusses the Results and the paper concludes in section V.

II. RELATED WORK

Reference [8] builds behavior graphs from IP addresses, port numbers, protocol and dependencies between network activities. Features are then extracted from these behavior graphs to be used for detection.

Among the features used for detection in approaches that use machine learning include portable PE header, API function calls [9], op-code sequences [10], system calls [11], TCP/IP packet header fields [12 & 13] and n-grams [14].

Common machine learning algorithms employed include ensemble models such as voting or cascading schemes [15], Perceptron algorithm to combine existing features, Restricted Boltzmann method for creating features for an increased detection rate [15 & 16], Hidden Markov Models (HMM) [10 & 11], structured multiclass SVM [17], Genetic Algorithms [12], Naive Bayes (NB) [14], OSC-3 [18], a combination of classifiers SVM, Rule Induction, kNN, NB, DT, ANN, Random Forest (RF) [15].

III. METHODOLOGY

As indicated in the Introduction, the main aim of this work is to investigate various machine learners on computer worm detection using unidirectional network traffic to a dark space. The methodology adopted will therefore follow the standard procedure in machine learning: 1) collecting data, 2) exploring and preparing the data, 3) training a model on the data, 4) evaluating model performance and 5) improving model performance.

Among the machine learning algorithms investigated included k Nearest Neighbors (kNN), Naïve Bayes (NB), Support Vector Machines (SVM), Neural Networks (NN) and Decision Trees (DT). Ensemble methods such as Random Forest (RF) will also be explored.

A. Dataset

The datasets used for the experiments were obtained from the University San Diego California Center for Applied Data Analysis (USCD CAIDA). The center operates a network telescope that consists of a globally rooted /8 network that monitors large segments lightly used address space. There is little legitimate traffic in this address space hence it provides a monitoring point for anomalous traffic that represents almost $1/256^{th}$ of all IPv4 destination addresses on the Internet.

Two sets of datasets were requested and obtained from this telescope. The first is the Three days of Conficker Dataset [19] containing data for three days between November 2008 and January 2009 during which Conficker worm attack [20] was active. This dataset contains 68 compressed packet capture (pcap) files each containing one hour of traces. The pcap files only contain packet headers with the payload having been removed to preserve privacy. The destination IP addresses have also been masked for the same reason. The other dataset is the Two Days in November 2008 dataset [21] with traces for 12 and 19 November 2008, containing two typical days of background radiation just prior to the detection of Conficker which has been used to differentiate between Conficker-infected traffic and clean traffic.

The datasets were processed using the CAIDA Corsaro software suite [22], a software suite for performing large-scale analysis of trace data. The raw pcap datasets were aggregated into the FlowTuple format. This format retains only selected fields from captured packets instead of the whole packet, enabling a more efficient data storage, processing, and analysis. The 8 fields are source IP address, destination IP address, source port, destination port, protocol, Time To Live (ttl), TCP flags, IP length. An additional field, value, indicates the number of packets in the interval whose header fields match this FlowTuple key. These features are further explained in section B below to motivate an understanding of their contribution towards the detection capability of the learning algorithms.

The instances in the Three Days of Conficker dataset have been further filtered to retain only instances that have a high likelihood of being attributable to Conficker worm attack of the year 2008. Ref. [20] focuses on Conficker's TCP scanning behavior (searching for victims to exploit) and indicates that it engages in three types of observable network scanning via TCP port 445 or 139 (where the vulnerable Microsoft software Windows Server Service runs) for additional victims. The vulnerability allowed attackers to execute arbitrary code via a crafted RPC request that triggers a buffer overflow.These are local network scanning where Conficker determines the broadcast domain from network interface settings, scans hosts nearby other infected hosts and random scanning. Other distinguishing characteristics of this worm included TTL within reasonable distance from Windows default TTL of 128, incremental source port, incremental source port in the Windows default range of 1024-5000, 2 or 1 TCP SYN packets per connection attempt instead of the usual 3 TCP SYN packets per connection attempt due to TCP's retransmit behavior.

This dataset solves the privacy challenge by removing the payload and also masking out the first octet of the destination IP address. It is also a more recent dataset than the KDD

dataset that has been the one available for network security researchers. However, it only includes unidirectional traffic to the network telescope and therefore does not allow the researcher to include features of computer worms that would be available in bidirectional traffic and would help with a more complete training.

B. Features

This section presents an analysis of the features to be used for detection and their contribution towards the detection capability of the learning algorithms.

Source IP address indicates the IP address of the originating host while Destination IP address indicates the recipient IP address. In the features to train the algorithms, the Destination IP address will be left out as it has been masked in the datasets and is therefore not useful in demarcating the classes. The source IP address can be used as a discriminating feature. Large variability in source IP address is unusual as hosts normally community with just a few hosts in normal communications. Particular geographical regions are also predisposed to more origin of computer worm attacks and this information can be obtained from the IP addresses. Reserved IP addresses, when seen as originating hosts are also suspicious.

Reflexive source and destination ports (similar) are also suspicious and can contribute to discrimination between classes. In addition, many computer worms target particular services whose ports are well-known and common. This can be a discriminant feature. For example, the Ramen worm uses port 21 while the Conficker worm uses ports 139 and 145. Source ports within particular ranges may also be indicative of computer worm activity.

Protocol indicates the next level protocol. ICMP is 1, TCP is 6 and UDP is 17. Worms can be classified based on the transport channel used. Even though this in itself cannot discriminate between classes, it can help limit the amount of traffic to deal with.

Time To Live (TTL) is used to avoid looping in the network Every packet is sent with some TTL value set, which tells the network how many network routers (hops) this packet can cross. At each hop, its value is decremented by one and when the value reaches zero, the packet is discarded. Different operating systems have default TTL ranges and since computer worms target vulnerabilities in particular operating systems, they will usually be associated with TTL within certain ranges.

Each TCP segment has a purpose and this is determined with the help of the TCP flag option. A value of 1 means that a particular flag is set. Flags occupy 6 bits. Each flag is 1 bit. The flags are URG, ACK, PSH, RST, SYN, and FIN [24]

URG: Urgent Pointer to identify incoming segment as urgent.

ACK: Acknowledgment used to acknowledge the successful receipt of packets.

PSH: PUSH to ensure that the data is given priority.

RESET: Used when a segment arrives that is not intended for the current connection, for example, if you were to send a packet to a host to establish a connection, and there was no such service waiting to answer at the remote host, then that host would automatically reject your request and then send you a reply with the RST flag set.

Packet length indicates the size of the packet. Particular computer worms are associated with particular packet length sizes. For example, the packet length for Conficker worm is around 62 bytes.

C. Machine Learning Algorithms

Various Machine Learning Algorithms were explored and their detection capabilities identified. There was an attempt to explain why they were successful in some cases and failed in others. The algorithms explored were the ones that have been commonly seen in the literature. These included k-Nearest Neighbors (kNN), Naïve Bayes (NB), Support Vector Machines (SVM), Decision Trees (DT), Artificial Neural Networks (ANN), and Random Forests (RF). The implementations of the algorithms used are as in the R programming language [16].

D. Experiments

Some flow tuples from the Conficker worm dataset were randomly selected and mixed with the some flow tuples from the clean traffic after each having been appropriately annotated into worm traffic and benign traffic. A random selection was then made to realize an eventual 1000 observables.

20% of the data was thereafter held back from the modeling process as the validation dataset to be used at the end of the project to confirm the accuracy of the final model. The remaining 80% was the reserved data for training and testing.

The variables were all numeric apart from the class variable which was nominal with 2 levels. A peek at the variables indicated that normalization of attributes would be necessary as the ranges varied greatly. Table 1 shows the frequencies and percentages of the composition of the training and testing dataset.

TABLE I. COMPOSITION OF TRAINING AND TESTING DATASETS

	Frequency	***Percentage***
Worm	480	59.92509
Not Worm	321	40.07491

Some modest correlation between IP packet length and protocol attributes was noticed at 0.78. Some learning algorithms would therefore benefit from removing the highly correlated attributes.

To further understand the data, visualizations were carried out using the R programming language [25].

For data pre-processing, data transforms were carried out to normalize the data. Observations with null values were weeded out and outliers also filtered out. A number of

transform functions were carried out, subjecting each to model training and prediction to be able to determine the best transform function for various learning algorithms.

For model training, various classifiers were explored. Since there was a good amount of data, 10-fold cross validation with 3 repeats was used. This is a good standard test harness configuration for binary classification problems. Accuracy and Kappa metrics were used as the model evaluation metrics.

The models were created initially using default parameters. Algorithm tuning was performed thereafter. The random number seed was set before training each algorithm to ensure each algorithm was evaluated on exactly the same splits of data. This would make later comparisons simpler.

Skewed distributions were thereafter adjusted using transform methods, starting with the box-cox transform.

The features were ranked in importance using R.

IV. Results and Discussion

A. Features

When the features were ranked, it was found out that they were in the order value, source IP address, Time To Live, IP Length, TCP flags, protocol, source port and lastly destination port as shown in Table II. It is evident that the three most useful features for the classification were value, source IP address and TTL. Destination Port provided the least classification ability.

TABLE II. FEATURE RANKING

	Importance
Value	0.7497
Src_ip	0.6163
TTL	0.5425
IP length	0.4967
Src port	0.3310
Dst port	0.1856

Figure I shows the ranking graphically. The feature value seems to most discriminating because it indicates the repletion of a particular tuple key. A number of flow tuples with a particular key would be suspicious. Also, the source IP addresses seems to be discriminative. It was our explanation that this could be because computer worm attacks origin seem to be localized to certain geographical regions. Time to Live is also ranked high. This may be because of the default Time to Live values for the attacked operating systems.

The metrics utilized for algorithm evaluation were Accuracy and Kappa metrics. The kappa statistic adjusts accuracy by accounting for the possibility of correct prediction by chance alone. Kappa values range to a maximum value of 1, which indicates perfect agreement between the model's predictions and the true values – a rare occurrence. It can be seen from the table that the kappa values show a very good agreement (values greater than 0.80). C5.0 decision tree algorithm again provides the highest kappa value at 0.9102. The algorithms performed as is depicted in Table III.

TABLE III. ALGORITHM PERFORMANCE (MEAN)

	Accuracy	***Kappa***
kNN	0.9289	0.8473
NB	0.8690	0.7141
C5.0	0.9575	0.9102
SVM	0.9325	0.8552
CART	0.9401	0.8730

Figure I show a plot of the algorithm performance in terms of accuracy and kappa. It can be seen that the C5.0 decision tree algorithm performed the best with an accuracy of 0.9575. It was therefore chosen as the best model and further tuned and utilized for prediction. The ConfusionMatrix for this algorithm is as shown in Table IV. A confusion matrix is a table that categorizes predictions according to whether they match the actual value in the data. The sensitivity of the model is 0.9916. Sensitivity measures the proportion of positive examples that were correctly classified. Over 99 percent of the positive class (worm) was correctly classified. The specificity of the model or the proportion of negative examples that were correctly classified is however at 0.8375. While this is still high, there is room for improvement. It indicates that almost 2 out of every 10 instances that are benign are wrongly classified as malicious.

TABLE IV. CONFUSION MATRIX

Sensitivity	0.9916
Specificity	0.8375

V. Conclusion and Future Work

The aim of the paper was to investigate the detective capability of machine learning algorithms in detecting computer worms in networks. The dataset used for the training and testing and eventual validation of the algorithms was the UCSD CAIDA network telescope darkspace traffic. Various machine learning algorithms were investigated such as Naïve Bayes, k Nearest Neighbors, Support Vector Machines, Decision Trees, etc. The implementations for these algorithms were those available in packages available to the R programming language. C5.0 decision tree algorithm emerged the best in terms of accuracy and confidence. The features that were useful for the detection capabilities were also determined. In future, the authors will investigate how to improve the detection using ensemble algorithms and even a

better understanding of the feature set from the same dataset.

REFERENCES

[1] N. Weaver , V. Paxson , S. Staniford , R. Cunningham, A taxonomy of computer worms, Proceedings of the 2003 ACM workshop on Rapid malcode, October 27-27, 2003, Washington, DC, USA [doi>10.1145/948187.948190]

[2] N. Ochieng, W. Mwangi, I. Ateya, A tour of the computer worm detection space, Internation Journal of Computer Applications, vol. 104, issue 1, pp. 29-33.

[3] S. Staniford, D. Moore, V. Paxson, N. Weaver, The top speed of flash worms, WORM 04 2004.

[4] R. Kaur, M. Singh, A survey on zero-day polymorphic worm detection techniques, IEEE Communications Surveys & tutorials, vol. 16. #3 3rd quarter 2014.

[5] A. Dainotti, A.Pescape and K.Claffy, Issues and future directions in traffic classification, IEEE Network, vol. 26, no. 1, pp. 35-40, January-February 2012.

[6] B. Tawfeeq, H. Qeshta, Adaptive worm detection model based on multi classifiers, Information and Communication Technology (PICICT), 2013 Palestinian International Conference on. IEEE, 2013.

[7] S. Robin, V. Paxson, Outside the closed world: On using machine learning for network intrusion detection, Security and Privacy (SP), 2010 IEEE Symposium on. IEEE, 2010.

[8] Nari, Saeed, and Ali A. Ghorbani. "Automated malware classification based on network behavior." Computing, Networking and Communications (ICNC), 2013 International Conference on. IEEE, 2013.

[9] D. Vatamanu, C., Cosovan, D., Gavriluţ, D., & Luchian, H. (2015). A Comparative Study of Malware Detection Techniques Using Machine Learning Methods. World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, 9(5), 1157-1164.

[10] G. Narra, U., Di Troia, F., Corrado, V. A., Austin, T. H., & Stamp, M. (2016). Clustering versus SVM for malware detection. Journal of Computer Virology and Hacking Techniques, 12(4), 213-224.

[11] Imran, Mohammad, Muhammad Tanvir Afzal, and Muhammad Abdul Qadir. "Similarity-based malware classification using hidden Markov model." Cyber Security, Cyber Warfare, and Digital Forensic (CyberSec), 2015 Fourth International Conference on. IEEE, 2015.

[12] Li, Wei. "Using genetic algorithm for network intrusion detection." *Proceedings of the United States Department of Energy Cyber Security Group* 1 (2004): 1-8.

[13] Shabtai, A., Moskovitch, R., Feher, C., Dolev, S., & Elovici, Y. (2012). Detecting unknown malicious code by applying classification techniques on opcode patterns. *Security Informatics*, *1*(1), 1.

[14] Lai, Yingxu, and Zhenghui Liu. "Unknown malicious code detection based on bayesian." *Procedia Engineering* 15 (2011): 3836-3842.

[15] Cimpoeşu, M., Gavriluţ, D., & Popescu, A. (2012). The proactivity of perceptron derived algorithms in malware detection. *Journal in Computer Virology*, 1-8.

[16] Benchea, R., & Gavriluţ, D. T. (2014, July). Combining restricted boltzmann machine and one side perceptron for malware detection. In *International Conference on Conceptual Structures* (pp. 93-103). Springer International Publishing.

[17] Mulay, Snehal A., P. R. Devale, and G. V. Garje. "Intrusion detection system using support vector machine and decision tree." *International Journal of Computer Applications* 3.3 (2010): 40-43.

[18] Barat, Marius, Dumitru Bogdan Prelipcean, and Dragos Teodor Gavrilut. "An Automatic Updating Perceptron-Based System for Malware Detection." *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2013 15th International Symposium on*. IEEE, 2013.

[19] The CAIDA UCSD Network Telescope "Three Days Of Conficker" - < dates used >,

http://www.caida.org/data/passive/telescope-3days-conficker_dataset.xml

[20] Emile Aben. Conficker/Conflicker/Downadup as seen from the UCSD Network Telescope. Technical report, CAIDA, February 2009. https://www.caida.org/research/security/ms08-067/conficker.xml

[21] The CAIDA UCSD Network Telescope "Two Days in November 2008" Dataset - < dates used >,

http://www.caida.org/data/passive/telescope-2days-2008_dataset.xml

[22] http://www.caida.org/tools/measurement/corsaro

[23] Alistair King. Corsaro. http://www.caida.org/tools/measurement/corsaro/, October 2012.

[24] https://tools.ietf.org/html/rfc793#page-15

[25] R Core Team (2013). R: A language and environment for statistical computing.Jul 16, 2013

[26] Shubair Abdulla, Sureswaran Ramadass, Altyeb Altaher and Amer Al-Nassir. Employing Machine Learning Algorithms to detect unknown scanning and email worms. The International Arab Journal of Information Technology, vol. 11, No. 2, March 2014.

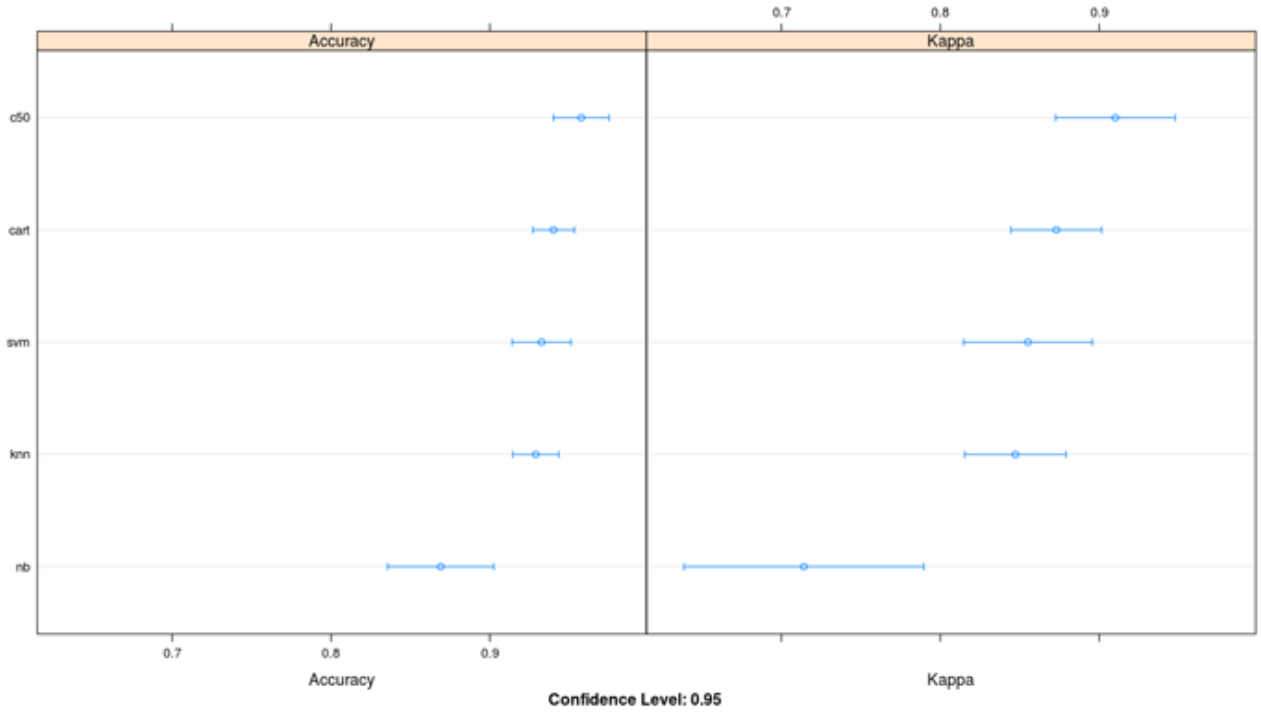

Fig. 1. Algorithms performance using the accuracy and kappa metrics