Hybrid Intrusion Detection System for Worm Attacks Based on Their Network Behavior

Hassan Hadi Latheeth AL-Maksousy(✉) and Michele C. Weigle

Department of Computer Science, Old Dominion University, Norfolk, VA 23529, USA
{halma002,mweigle}@odu.edu

Abstract. Computer worms are characterized by rapid propagation and intrusive network disruption. In this work, we analyze the network behavior of five Internet worms: Sasser, Slammer, Eternal Rocks, WannaCry, and Petya. Through this analysis, we use a deep neural network to successfully classify network traces of these worms along with normal traffic. Our hybrid approach includes a visualization that allows for further analysis and tracing of the network behavior of detected worms.

Keywords: Deep learning · Worm traffic · Internet worms · Sasser Slammer · NotPetya · WannaCry · EternalRocks · Visualization

1 Introduction

Computer worms are small self-duplicating code that can rapidly infect hundreds of thousands of systems. The Slammer worm spread to 90% of its target systems in just 10 min [1]. In 2017, the NotPetya and WannaCry worms created a global panic [2]. A great deal of research based on different frameworks and numerous case studies are available for the analysis of the spread of worms across networks [3–6]. Worms' detrimental influence on computer systems has been significant over the last several decades. The worms propagate from one machine to the next and are detected using different methods. Therefore, it has become important to develop intrusion detection systems and prevention mechanisms to counter them.

The main goal of this work is to detect and visualize worms by understanding their network behavior. Our approach is a hybrid system that consists of machine learning intrusion detection and a visualization tool in order to classify and detect worms based on their network behavior. Our hybrid approach is meant to confront and deal with anomaly and network behavior aspects of worms through the use of machine learning and a visualization tool. This study implements a network traffic generator using the NS3 simulator, a packet analysis tool for examining PCAP files, a deep neural network approach to classify and detect worms, and a visualization tool based on D3 for analyzing network traces.

F. Breitinger and I. Baggili (Eds.): ICDF2C 2018, LNICST 259, pp. 225–234, 2019.
https://doi.org/10.1007/978-3-030-05487-8_12

2 Related Work

Signature-based detection uses a database of signatures containing known worms [7–9]. The detection algorithm uses this database to determine if a packet is infected, i.e., if it contains a known signature of a worm in the database. The main drawback of this method is that it relies upon knowing these signatures, therefore new worms will go undetected until they are discovered and added to the database. Due to the limitations of signature-based detection, anomaly or behavior-based detection was introduced. This approach looks for anomalous behavior of a worm by distinguishing it from what is normal behavior [10,11]. This function and purpose of a worm characterizes its behavior and provides a profile for this method.

Several recent works have analyzed network behavior for worm detection and classification. Sarnsuwan et al. [12] use network-based Internet worm detection utilizing thirteen packet features. Data mining algorithms such as Bayesian Network, Decision Tree, and Random Forest are used for classification of Internet worms, normal data, or network attack data. Barhoom et al. [13] propose a model that makes use of data mining techniques by combining different classifiers with an adaptive approach for detecting known or unknown worms. Another technique proposed by Rasheed et al. [14] focuses on detecting polymorphic worms. However, only a single worm MSblaster was used for testing. Tang et al. [15] proposed an intrusion detection system using the Deep Neural Network (DNN) model. A visualization of the computer system state during a worm attack requiring manual intervention by the user is presented by Axelsson et al. [16]. It was observed that the worm request clusters were noticeably different from the clusters formed by normal traffic.

3 Analyzed Worms

We analyze and develop a Finite State Machine (FSM) model for the following worms: Sasser, Slammer, Petya, WannaCry, and EternalRocks.

3.1 Sasser Worm

The Sasser worm targets computers running Microsoft Windows XP and Windows 2000. It has the capability of spreading rapidly over vulnerable computers via TCP port numbers 445 or 139 to infect other computers without any human interaction. This worm belongs to self-replicating worms from the W32.Sasser family [17] and exploits a vulnerable LSASS.EXE to infect the system, allowing the attacker to gain full control of the system [18].

3.2 Slammer Worm

The Slammer worm exploits a buffer overrun in the Microsoft SQL service. The worm's payload is small enough to fit inside a single packet. It spreads very rapidly [19] infecting unpatched SQL servers. It propagates by transmitting the same message to other SQL servers rapidly using random IP addresses. This attack can only be mitigated by taking the server offline. There is no additional malicious content included in the worm payload. However, because of the behavior and the speed with which it attacks systems, it executes an effective DoS attack as the network's resources are drained [20].

3.3 EternalRocks, WannaCry and Petya

EternalRocks, WannaCry and Petya utilize the EternalBlue exploit which was leaked as part of the EternalRocks Shadow Brokers dataset[1]. EternalBlue exploits a flaw in the SMBv1 service's NT Transaction request handler by sending a large NT Transaction Request, making it necessary for the target to accept one or more Trans2 Secondary requests. By sending malformed Trans2 Secondary requests, EternalBlue exploits a flaw in the protocol that allows shellcode to be delivered and executed on the target machine [2].

4 Approach

The machine learning approach is employed to classify and detect predefined worms and undefined worms with similar network behavior. Once the machine learning tool detects suspicious activity, then the visualization tool will be used to further analyze the network traffic. In analyzing the traffic, the visualization tool displays suspicious worm activity, traces the attack, and displays the affected nodes.

Our system architecture is shown in Fig. 1. Using a PCAP file as an input to the ipsumdump utility [21], we extract features as a CSV file. The preprocessor converts the features into a numerical array and normalizes the data. The data is passed to the deep neural network (DNN) for detection of worms. If the machine learning tool detects suspicious activity, the visualization tool is used to display the suspicious worm activity, trace the attacks, and display the affected nodes.

We use the following features from ipsumdump:

- wire-length: Overall length of captured packet, including headers
- length: IP packet length
- protocol: IP protocol
- ip-id: IP-ID field
- ip-sum: IP checksum
- ip-ttl: IP time to live field
- ip-tos: IP type of service field
- ip-hl: IP header length
- capture-length: Length of IP data
- sport: Source TCP or UDP port
- dport: Destination TCP or UDP port

[1] http://www.ericconrad.com/2017/04/shadowbrokers-pcaps-etc.html.

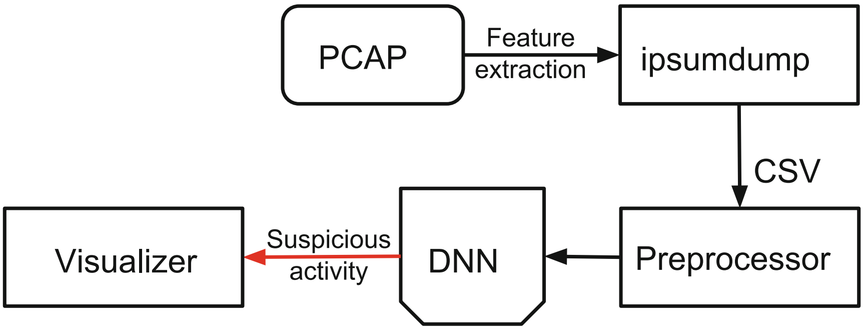


Fig. 1. System architecture

- payload-md5-hex: MD5 checksum of payload
- tcp-seq: TCP sequence number
- tcp-ack: TCP acknowledgement number
- tcp-window: TCP receive window
- udp-length: UDP length, reported in the header
- icmp-type: ICMP type
- icmp-code: ICMP code

4.1 Machine Learning

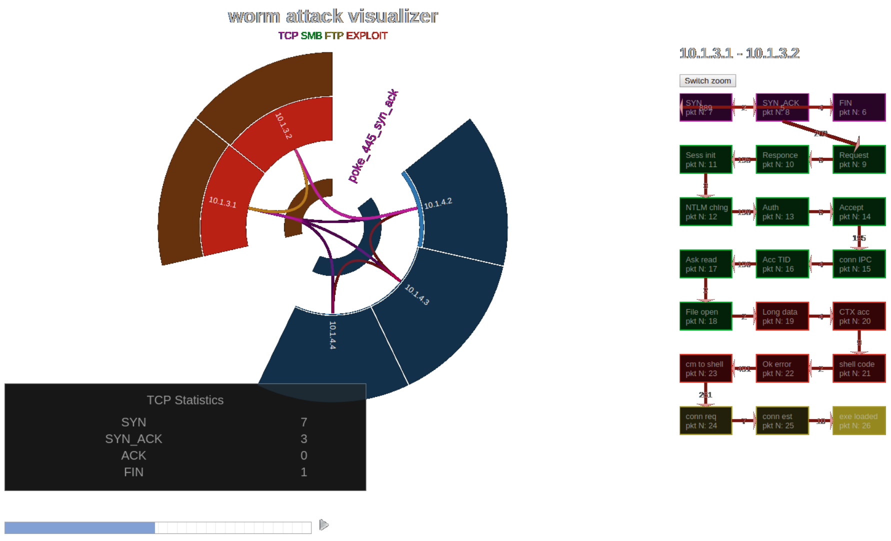
Deep neural network is a machine learning model that is trained using the dataset. DNNs are used to solve complex problems and possess the ability for unsupervised learning, a key component for automatically detecting worm variants and mutations.

We used the framework of Keras [22] to build a fully connected DNN. As a proof of concept, we trained the DNN model with five real time PCAP capture files from online sources [23–25] containing captures of worm traffic. In addition to the worm traffic, we also captured normal network traffic generated by our machine. The overall size of the processed dataset for training was 11.2 MB of malicious traffic and 85.1 MB of normal traffic. The test dataset was 10% of the entire dataset.

4.2 Worm Visualization

One of the main contributions of this work is building the visualizer IDS with D3. D3[2] is a JavaScript library that is used to create data visualizations. The data generated by computer networks is considerably large and visualization methods such as D3 are necessary to interpret and process this overwhelming data. The visualization tool is designed to help interpret the large amount of data and display this data on the screen in a visually appealing format.

[2] https://d3js.org.



TCP Statistics

SYN	7
SYN_ACK	3
ACK	0
FIN	1

Fig. 2. Worm visualization. The FSM of the detected worm is shown on the right.

The worm visualization analysis tool presented in this paper uses the Finite State Machine (FSM) model, which defines the expected behavior of worm network traffic. When the behavior deviates from this expectation, the FSM will not display the new states, however, we will still be able to detect the worm attack using machine learning. Our packet analysis tool processes one or more captured network PCAP files and produces a JSON file. The JSON file is provided to the visualization tool to be interpreted. We use our FSM model to divide the packets into four categories: TCP, SMB, FTP, and Exploit. Packets in the Exploit category are various worm states of attack. Four colors have been assigned to these categories in order to visually determine the kind of packet. Exploit packets are red, and any red activity in the display can be associated with potential or actual worm attacks.

An example of the visualization tool can be seen in Fig. 2. The visualization tool displays each node represented by a wedge shape. Nodes are grouped based on their IP address and a specific color is assigned based on the network the node belongs to. The wedge shape is then filled depending on the overall percentage of the defined packet activity. This makes it easier to observe the overall activity distribution on the nodes during an analysis session. However, if a worm attack is detected, the activity area will be displayed as red. Packet transmissions between any two nodes are depicted by an arc made in the color of the packet category. Since the visualizer shows the network traffic over time, these interconnecting arcs will change and fade color as other categorized packets are transmitted. The complete PCAP session, whether from one or multiple files, is combined into a single timeline. We can navigate from the beginning to the end and to any point along the timeline of the session. The visualizer will update to show all activity up to the current index. Along with the pictorial representation of inter-connectivity and packet activity, the visualization tool displays packet statistics. At the center of the main graph the current packet type is always displayed. This text continuously and quickly updates based on the current timeline position. There is an overview of general TCP statistics, showing different TCP packet types and counts for each type. This is also dynamically updated with the position of the timeline. When a node is clicked, detailed statistics for that node is displayed, as seen in Fig. 2. When clicking on the connection between two nodes, we can see detailed FSM information. As with all parts of this tool, these statistics update dynamically based upon the current timeline position. This detail helps us determine in which direction the infection is occurring. From this view, we can click on a specific packet and see even more detail about its state. By changing the index on the timeline, it is easy to determine which node gets infected first by seeing the first node turn red, we can then determine the source of the attack.

5 Evaluation

5.1 Worm Traffic Generator Using NS3 Simulator

NS-3[3] is an open-source discrete event network simulator. Using Wireshark, a packet capture and analysis tool, we gather information about a particular worm. This information is fed into the simulator to simulate an attack from the worm. We simulate an attack because it is very difficult to get all attack data from online sources as many companies hide details of the attack for security reasons. Due to the hiding of the data, we cannot see the propagation of the worm throughout the network. Using a test network and the simulator allows us to simulate an attack and provides us with complete data about the attack giving a clear picture of the network behavior of the worm. In addition, by using the traffic generator we can change the network topology and simulate multiple worm attacks, capturing the attacks to PCAP files.

The network simulator configuration consist of six nodes in total as shown in Fig. 3. Nodes 0 and 1 are in Subnet 1. Nodes 3, 4, and 5 are in Subnet 2. Node 2 is a router that exists in both subnets and routes the traffic through them.

The generated traffic consists of normal traffic, SMB, FTP, and multiple packets containing the behavior of the Sasser and Slammer worms.

5.2 Experiment

We performed four experiments to demonstrate the ability of our method to analyze network traffic.

In the first experiment, we analyze the performance of our DNN to classify network traffic as being part of Sasser, Slammer, Eternal Rocks, Wannacry, or Petya worm attacks or as being normal traffic. Recall that our dataset, described

[3] https://nsnam.org.

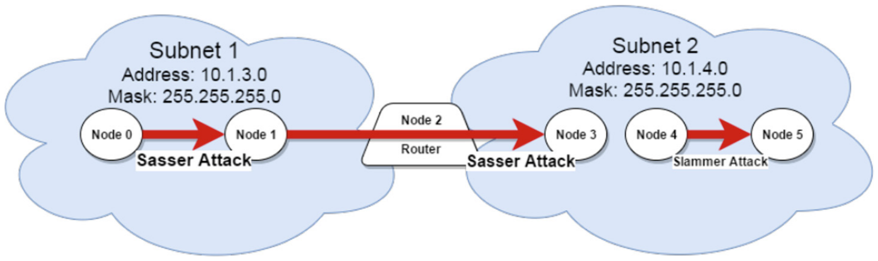


Fig. 3. Simulated evaluation network

Table 1. Performance of the DNN on classifying worm and normal traffic with known worms.

Name	Precision	Recall	F1
Sasser	0.8387	0.8125	0.8254
Slammer	1.0000	1.0000	1.0000
EternalRocks	0.9650	0.8894	0.9257
Wannacry	0.9581	0.9970	0.9772
Petya	0.9444	0.3119	0.4690
Normal	0.9985	0.9998	0.9991

in Sect. 4.1, had 11.2 MB of worm traffic and 85.1 MB of normal traffic. We used 90% of the dataset for training and 10% of the dataset for testing. Table 1 shows the precision, recall, and F1 score for the classification task on the test dataset. Our DNN achieved high precision in most cases. The precision for Sasser was lower, likely due to its high similarities to other worms. We observed low recall for the Petya worm, likely because its sample included normal traffic packets. We note that our classification of normal traffic was at 99%, which indicates that our system would have relatively low false alarms.

In the second experiment, we tested the system using a PCAP file containing both Sasser and Slammer worm traffic, which was generated using our NS-3 worm traffic generator tool. The PCAP file was applied as the input to the detection system and yields the results shown in Table 2. Our system uses packet by packet analysis to detect worms. The probability is calculated as:

$$P_n = \frac{\Sigma V[n]}{N}, \tag{1}$$

where $V[n]$ is the vector of probabilities generated by DNN and N is the number of vectors. In cases where the worm consists only of a single packet, the probability is low. In other cases where the worm attack consists of multiple packets, the probability is high. Despite the Slammer attack consisting of only one packet and having low probability, our system was still able to detect it.

Table 2. DNN classification probabilities on the simulated Sasser and Slammer worms.

Name	Probability
Sasser	0.7679
Normal	0.2030
Slammer	0.0012

Table 3. DNN classification probabilities on the unknown NotPetya sample.

Name	Probability
Petya	0.6046
Normal	0.2275
WannaCry	0.1206

In the third experiment, we wanted to test the system on a previously unseen, but similar worm. The main advantage of a DNN-based analyzer is the possibility to detect worms that may not be explicitly present in its training dataset. We obtained a sample of the NotPetya worm from CTU captures [26] and tested it with our classifier. NotPetya has similar, but not exactly the same, network behavior as the Petya worm. Table 3 shows that the DNN reported a 60% probability that the NotPetya trace was the Petya worm, which shows that the DNN can detect worms with similar network behavior.

Finally to further validate our DNN, we classified a sample of normal traffic obtained from Netresec[4]. Our DNN reported that the traffic was normal with a 99.996% probability, successfully classifying the normal traffic.

6 Conclusion

In this work, we have presented a hybrid worm detection and analysis approach. We trained a DNN classifier on the network traffic produced by several worms and normal network traffic. On the five worms tested, our classifier had an average precision of 94.1%. We also successfully classified a set of normal traffic with a 99.99% probability. In addition, we used our trained classifier to detect worm traffic that was unknown to the classifier but had network traffic similar to one of the known worms. The second part in our hybrid approach was the development of a visualizer based on D3 to observe the network and infection behavior of identified worms and to trace the attacks. The proposed hybrid approach provides better insight on worm activities and better detection of worms that have similar network behavior.

[4] https://www.netresec.com/?page=PCAP4SICS.

References

1. Moore, D., Paxson, V., Savage, S., Shannon, C., Staniford, S., Weaver, N.: Inside the slammer worm. IEEE Secur. Priv. **1**(4), 33–39 (2003)
2. Islam, A., Oppenheim, N., Thomas, W.: EternalBlue SMB protocol exploit (2017). https://www.fireeye.com/blog/threatresearch/2017/05/smb-exploited-wannacry-useof-eternalblue.html
3. Mishra, B.K., Jha, N.: SEIQRS model for the transmission of malicious objects in computer network. Appl. Math. Model. **34**(3), 710–715 (2010)
4. Mishra, B., Pandey, S.: Fuzzy epidemic model for the transmission of worms in computer network. Nonlinear Anal. R. World Appl. **11**(5), 4335–4341 (2010)
5. Toutonji, O.A., Yoo, S.-M.: Stability analysis of VEISV propagation modeling for network worm attack. Appl. Math. Model. **36**(6), 2751–2761 (2012)
6. Li, P., Salour, M., Xiao, S.: A survey of internet worm detection and containment. IEEE Commun. Surv. Tutor. **10**, 20–35 (2008)
7. Tang, Y., Chen, S.: Defending against internet worms: a signature-based approach. In: Proceedings of IEEE INFOCOM, vol. 2, pp. 1384–1394, March 2005
8. Kim, H.A., Karp, B.: Autograph: toward automated, distributed worm signature detection. In: Proceedings of the USENIX Security Symposium (2004)
9. Al-Hammadi, Y., Leckie, C.: Anomaly detection for internet worms. In: Proceedings of the 9th IFIP/IEEE International Symposium on Integrated Network Management, pp. 133–146, May 2005
10. Lakhina, A., Crovella, M., Diot, C.: Mining anomalies using traffic feature distributions. In: Proceedings of ACM SIGCOMM, pp. 217–228 (2005)
11. Chen, X., Heidemann, J.: Detecting early worm propagation through packet matching. Technical report, University of Southern California, Information Sciences Institute (2004)
12. Sarnsuwan, N., Charnsripinyo, C., Wattanapongsakorn, N.: A new approach for internet worm detection and classification. In: INC2010: 6th International Conference on Networked Computing, pp. 1–4, May 2010
13. Barhoom, T.S., Qeshta, H.A.: Adaptive worm detection model based on multi classifiers. In: Palestinian International Conference on Information and Communication Technology, pp. 57–65, April 2013
14. Rasheed, M.M., Badrawi, S., Faaeq, M.K., Faieq, A.K.: Detecting and optimizing internet worm traffic signature. In: 8th International Conference on Information Technology (ICIT), pp. 870–874, May 2017
15. Tang, T.A., Mhamdi, L., McLernon, D., Zaidi, S.A.R., Ghogho, M.: Deep learning approach for network intrusion detection in software defined networking. In: International Conference on Wireless Networks and Mobile Communications (WINCOM), pp. 258–263, October 2016
16. Axelsson, S.: Visualization for intrusion detection-hooking the worm. In: 8th European Symposium on Research in Computer Security ESORICS 2003 (2003)
17. Malaspinas, S.: Getting Sassy with Microsoft - An in depth analysis of the LSASRV.dll vulnerability. Global Information Assurance Certification Paper, pp. 1–56 (2004)
18. Abrams, T.: Microsoft LSASS buffer overflow from exploit to worm. SANS Network Security (2004)
19. Zheng, H., Lifa, W., Huabo, L., Fan, P.: Worm detection and containment in local networks. In: International Conference on Computer Science and Information Processing (CSIP), pp. 595–598 (2012)

20. Dübendorfer, T., Wagner, A., Hossmann, T., Plattner, B.: Flow-level traffic analysis of the blaster and sobig worm outbreaks in an internet backbone. In: Julisch, K., Kruegel, C. (eds.) DIMVA 2005. LNCS, vol. 3548, pp. 103–122. Springer, Heidelberg (2005). https://doi.org/10.1007/11506881_7
21. IPSumDump (2004). http://read.seas.harvard.edu/~kohler/ipsumdump
22. Francois Chollet. Keras (2015). https://keras.io/
23. Sasser (2017). https://wiki.wireshark.org/SampleCaptures
24. Wannacry (2017). https://precisionsec.com/wannacry-pcap-smb-445/
25. EternalRocks (2017). https://github.com/stamparm/EternalRocks/blob/master/misc/exploitation.pcap
26. CTU Malware Capture Botnet (2017). https://mcfp.felk.cvut.cz/publicDatasets/CTU-Malware-Capture-Botnet-288-1/