Azab, Ahmad, Alazab, Mamoun and Aiash, Mahdi ORCID logoORCID: https://orcid.org/0000-0002-3984-6244 (2016) Machine learning based botnet identification traffic. 2016 IEEE Trustcom/BigDataSE/ISPA. In: 15th IEEE International Conference on Trust, Security and Privacy in Computing and Communications (IEEE TrustCom-2016), 23-26 August 2016, Tianjin, China. ISBN 9781509032051. ISSN 2324-9013 [Conference or Workshop Item] (doi:10.1109/TrustCom.2016.0275)

Final accepted version (with author's formatting)

This version is available at: https://eprints.mdx.ac.uk/21927/

ACCEPTED MANUSCRIPT

Ahmad Azab (Telstra, Australia); Mamoun Alazab (Macquarie Univ., Sydney, NSW, Australia); Mahdi Aiash (Middlesex Univ., London, UK), Machine learning based Botnet Identification Traffic. 2016 IEEE TrustCom-BigDataSE-ISPA

SECTION I.

Introduction

Botnet is defined as a collection of compromised computers that are remotely controlled by a Botmaster, through a Command and Control (C&C) entity. The access to these computers is enabled after installing a malware, which creates a backdoor entry point for full remote access without the user's knowledge. The main difference between botnet and other types of malware is the use of the C&C infrastructure, which helps the Botmaster to manage the infected devices remotely and to conceal his/her identity.

Botnets are currently the main malware platform threat used by cybercriminals and the cause of most internet security threats [1] [2]. The research by Stevanovic, et al. [3] stated that different studies showed that 40% of computers in globe are part of botnets. Microsoft stated that 6.5 million infected devices with bots malware were treated and cleaned in 2010 [4]. Also, Symantec stated that on average, 42 billion spam messages a day around the world were caused by botnets [5]. Furthermore, statistics showed that more than 83% of spam emails around the world were caused by botnets in 2011 and 600% population growth of infected devices with bot malware was noticed in 2010 [6].

Solutions have been proposed to identify the C&C channel traffic between the infected devices and the C&C server in order to identify the infected devices, take the C&C server down, tracking down cybercriminals and to provide a solution that does rely on neither infection nor attack phases. The literature addressed the solutions in terms of, online classification capability by monitoring a few packets in a flow, supporting various transport and application protocols as TCP, UDP, HTTP and IRC, avoiding accessing packet's content, relying on a single phase's traffic for the detection process by monitoring only the C&C channel traffic, monitoring a single device's network traffic, detecting untrained versions of a botnet by analysing and building the classifier on a different single version.

Although these characteristics were fulfilled by some proposed approaches by researchers in the literature, no previous study attained all of these characteristics in a single solution. To fill this current gap in the literature, the novel methodology Classification of Network Information Flow Analysis (CONIFA), which has been proposed in our earlier work for detecting Skype new version's traffic [7], is used to detect a new version of a botnet by analysing and building the classifier on an old version. For this research study, Zeus botnet is deployed to address the effectiveness in applying CONIFA to detect its untrained version.

The rest of the paper is organized as follows. Section II introduces Zeus botnet. In section III, the literature review in detecting botnet C&C network traffic is provided. The datasets collection for Zeus different versions and non-Zeus network traffic and the extraction of network features are explained in section IV. In section V, the experimental results in applying the standard machine learning and CONIFA frameworks in detecting Zeus C&C traffic are listed. A discussion of the results is given in section VI, highlighting the efficacy of CONFIA in detecting a new, untrained, version of

Zeus is given. Finally, conclusions for this work along with some future directions are provided in section VII.

SECTION II.

Zeus Overview

Zeus, or Zbot, is a toolkit crime that emerged in 2007 and aims to implement malware that is used in a botnet attack and manages the zombies in that botnet. Its functionality is to steal personal credentials, especially online bank accounts credentials [8] for Windows OS. Symantec described Zeus toolkit as the "king of the Underground Crime ware Toolkits" [9]. In May 2011, Zeus V2 source code was publically leaked. The leakage allowed different malware creators to use the source code to implement a large number of new versions and derivatives with high complexity and new functionalities [10]. Such versions are subversions of Zeus version 2, and derivatives as Citadel and P2P game over Zeus. The prevalence of new versions and derivatives of Zeus in a rapid manner raised a major security threat in the ability to detect them with a low resnorise time to *reduce their* impacts.

In 2009, Zeus reached 3.6 million infections in the U.S alone [1]. In 2011, Zeus was reported as the most used online banking attack malware responsible for 80% of total attacks and revenue of $1 billion in the last five years globally [11]. Celik, et al. [12] reported that 6.2% of the total malware attacks were financial attacks with an increase of 1.3% in 2013 compared to 2012. These attacks were mainly conducted by Zeus. According to a recent study by Dell SecureWorks Counter Threat Unit (TM) (CTU), financial attacks targeted almost all types of financial institutions, mostly on traditional banking websites, and were mainly against institutions in the U.S [13]. The attacks were distributed across 65 countries, but attack rates increased in Asia, Africa and the Middle East. It was noticed that Zeus malware, with its derivatives Gameover and Citadel, are the most common crime toolkits for financial attacks.

SECTION III.

Related Work

Deep Packet Inspection (DPI) identifies C&C channels through defining signatures by conducting intensive analysis of the C&C channels packet's content. Binkley and Singh [14] proposed a DPI approach to detect IRC C&C channels by monitoring both TCP SYN scanning traffic and IRC channels traffic. Domain Name Server (DNS) approach detects botnets C&C channels by identifying DNS traffic to malicious domains. The definition of a malicious domain is conducted by defining different characteristics that distinguish malicious DNS request behaviour from benign DNS request behaviour. Hyunsang, et al. [15] identified five cases when botnet's DNS queries are issued: as soon as the device is iefected, if a malicious attack is conducted, C&C server failure, moving to different C&C servers and changing C&C IP address.

Temporal approaches detect C&C channel traffic by monitoring the temporal behaviour for the monitored device's network traffic when communicating with a destination end, by the fact that infected devices communicate with a C&C server over regular or irregular intervals. Inspired by the persistency concept [16], the authors Fedynyshyn, et al. [17] proposed a solution to use the persistency to detect C&C channels and classify them to their architecture (IRC, HTTP or P2P) by monitoring individual host's traffic. Garasia, et al. [18] applied the Apriori association algorithm [19] to identify the presence of a C&C channel for HTTP botnets, by applying four main phases named traffic representation, filtering, separation and detection. Correlation approaches identify C&C

channel traffic by relying on the fact that different bots within the same botnet generate similar network traffic characteristics for C&C and/or attack type. It aims to group hosts that share the same traffic characteristics to a destination domain and/or share the same attack type. Botminer [20], an extension of Botsniffer [21], identifies botnet activity by correlating bots with the same C&C channel traffic and attack traffic by proposing five main phases. Zang, et al. [22] proposed hierarchical and k-mean clustering for detecting botnets C&C traffic.

Machine learning approach, flows classification using machine learning algorithms, characteristics (features) that are able to distinguish the C&C flows from legitimate flows are identified and used to build a classifier model along with classification algorithms to detect the C&C flows. Kirubavathi, and Anitha [23], [24] built a classifier to detect C&C channel by extracting features from a host traffic to a destination end for a defined interval deploying various learning algorithms. Utilizing C4.5 and GP algorithms, Haddadi, et al. [25] classified HTTP C&C channel traffic. Detecting botnet traffic online by monitoring a number of packets in a flow was addressed by Stevanovic and Pedersen [26]. The authors Zhao, et al. [27] addressed the ability to detect botnet traffic by monitoring a small part of a flow and to detect unknown botnets by building a classifier on known botnets. The authors in [28] and [31] proposed a behavioural approach to detect Malware.

SECTION IV.

Datasets

Datasets need to be collected in order to evaluate the proposed methodology, CONIFA, and the standard machine learning framework used by researchers in the literature in where they should contain both botnet and non-botnet network traffic. Two datasets, training and testing, were collected. The training dataset contains botnet traffic for the same version and non-botnet traffic. The testing dataset contains botnet traffic, for a different version from the one used in the training dataset, and non-botnet traffic. For botnet network traffic representation, Zeus was chosen since, as illustrated earlier, it is one of the major threats, especially for online bank attacks. As the main goal of this experiment is to address the ability to detect the untrained version, Zeus builder's versions 1.2.7.11, 1.2.7.19, 1.3.1.1, 2.0.8.9 and 2.1.0.1 were used to generate the network traffic. The traffic generated using version Lx is defined as the trained version where the traffic generated using version 2.x is defined as the untrained version for the conducted experiment.

For this experiment, we have collected only the HTTP GET connection to increase the chances to identify the infected device before uploading/stealing any data from it. Network traffic was collected by capturing all the packets entering/leaving the network card and stores them in the form of PCAP file. The two devices (the C&C server and the infected device) were connected to two different networks to provide the real life scenario's traffic. For non-botnet traffic, we emulated the real life scenario by collecting benign HTTP traffic from two sources. The first source was accessing various websites and collecting their HTTP network traffic in our lab. The top 23 websites listed in Alexa were accessed and their network traffic were collected.

The second source of the HTTP traffic was downloading different files from the internet with various sizes. The aforementioned sources ensure the variety of the collected HTTP network traffic in which they replicate the use of the internet in the real life scenarios by different users.

The collected datasets, training and testing, are illustrated in Table I, in terms of the number of the flows. The training dataset comprises Zeus version 1.x and non-Zeus flows whereas the testing dataset comprises Zeus version 2.x and non-Zeus flows.

In order to build a classifier to detect botnet C&C traffic using the standard framework used by researchers in the literature or by our framework CONIFA, statistical features should first be identified. The mining of statistical features is conducted by applying deep analysis for the collected network traffic. Aside from the network analyst's expertise, different tools such as Wireshark are used to help the network analyst to extract and identify different features of the collected network traffic. The analysis was conducted by monitoring the C&C packets behaviour over time within a flow as well as checking the header's values, and extracted features that express their behaviour and help in distinguishing them from the benign flows. The identified features with a brief description for each one are to be found in [29].

Netmate and Wireshark were used in order to extract these statistical features from our datasets over unidirectional and bidirectional flows. The TCP flow is terminated either if a flow tears down or hits the threshold (600 seconds).

SECTION V.

Experimnetal Results

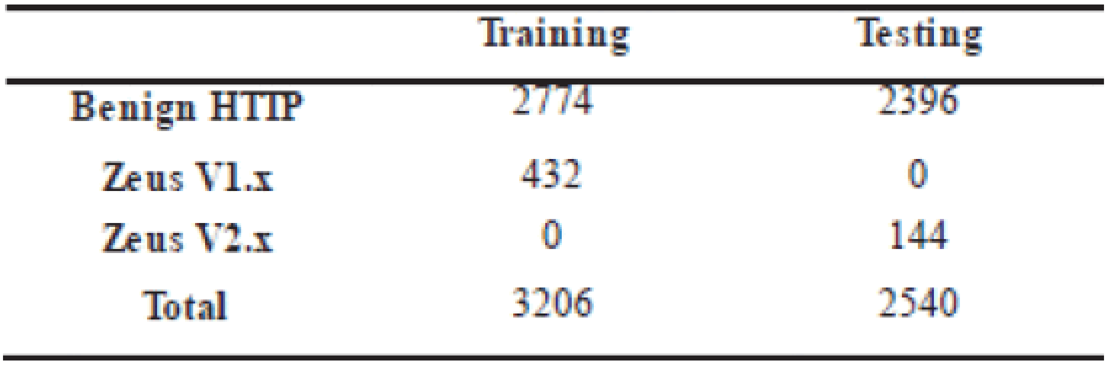
C4.5 learning algorithm and Correlation-based Feature Selection (CFS) algorithm were used in this experiment as they showed their efficacy in detecting the untrained version of Skype [7]. For the evaluation purposes, Precision, Recall and F-Measure metrics were chosen as our dataset is imbalanced and they provide accurate measures for imbalanced datasets [30]. Precision reflects the accuracy of the flows that were classified as Zeus by the classifier. It reflects if the flows identified as Zeus were actually Zeus. Recall reflects the accuracy in classifying Zeus flows from the entire dataset. F-Measure is defined as the weighted average of both Recall and Precision.

A. The Standard Framework

In this subsection, the standard framework is evaluated in terms of identifying Zeus versions 1.x and 2.x by building the classifier using Zeus version 1.x

The identified features were extracted from the flows of the collected training dataset. After that, CFS feature selection algorithm was used to filter out redundant and irrelevant features. The selected nine features were MPF (Mean Packets Forward), LPF (Largest Packet Forward), STDTF (Standard Deviation inter-arrival Time Forward), STB (Minimum interarrival Time Backward), MTB (Mean inter-arrival Time Backward), STDTB (Standard Deviation inter-arrival Time Backward), SActive (Minimum flow Active), LIdlc (Maximum flow Idle) and BPush (Number Push Backward).

Table I: The two datasets collected for the zeus experiment



	Training	Testing
Benign HTIP	2774	2396
Zeus V1.x	432	0
Zeus V2.x	0	144
Total	3206	2540

C4.5 cost insensitive algorithm was deployed to build the classifier, utilizing the training dataset defined in Table I, with the aforementioned nine features. The built classifier is evaluated first for detecting Zeus flows for the same version used in building the classifier and second for detecting the untrained version that was not used in building the classifier.

For the first evaluation, the built classifier was evaluated using the K-10 cross validation. The classifier provided, as shown in Fig. 1 a 1.000 Recall result, indicating that all Zeus trained version flows were successfully identified. This reflects that FN result was 0. For the Precision, the built classifier provided 0.993, which reveals that the flows identified as Zeus were mainly Zeus. This is because 3 non-Zeus flows were misclassified as being Zeus. The F-Measure scored 0.997, reflecting that the built classifier provided high accuracy results in terms of identifying Zeus flows from the entire dataset and flows identified as Zeus were truly Zeus.

For the second evaluation, the built classifier was evaluated using the testing dataset defined in Table I, along with the aforementioned nine features. The Recall result was 0.556, indicating that almost half of Zeus flows from the entire dataset were not identified. This is because the FN result was 64. The Precision scored 0.964, reflecting that the flows identified as Zeus were mainly Zeus. 3 non-Zeus flows were misclassified as being Zeus. The F-Measure scored 0.705, reflecting that the built classifier provided acceptable accuracy results in terms of identifying Zeus flows from the entire dataset and flows identified as Zeus were truly Zeus. These results indicate that the standard framework performance does degrade in detecting the untrained version's traffic for Zeus malware compared to detecting the trained version.

The evaluations showed that deploying the standard framework to identify a trained version's flows of Zeus provided high accuracy results. However, this approach's performance does degrade in identifying the untrained version's flows of Zeus malware as not all features have the same values for different versions.

B. Conifa Framework

The identified features were extracted from the flows of the collected training dataset. After that, CFS feature selection algorithm was used to filter out redundant and irrelevant features, resulting of the nine features mentioned earlier. For the first step, all the possible feature combinations for the resulted features were 511 combinations. For these combinations, C4.5 cost sensitive algorithm was used with 10, 20 and 30 FN costs and 1 FP cost to build the classifiers, utilizing the training dataset defined in Table I.

For the second step, all the built classifiers were evaluated utilizing the K-10 cross validation to choose the lenient classifiers. For cost 10, the lowest FN with the highest FP results were 0 and 34 respectively and were chosen as the lenient classifier, reflecting that all Zeus flows were identified and 34 non-Zeus flows were misclassified as being Zeus. For cost 20, the lowest FN with the highest FP results were 0 and 35 respectively and were chosen as the lenient classifier, reflecting that all Zeus flows were identified and 35 non-Zeus flows were misclassified as being Zeus. For cost 30, the lowest FN with the highest FP results were 0 and 36 respectively and were chosen as the lenient classifier, reflecting that all Zeus flows were identified and 36 non -Zeus flows were misclassified as being Zeus. The selected classifier's feature combination for the three costs was LPF, STB and LIdle.

For the third step, C4.5 cost sensitive algorithm was used with 10, 20 and 30 FP costs and 1 FN cost to build the classifiers for the *51*1 combinations identified earlier, utilizing the training dataset defined in Table I.

Fourth, the built classifiers were evaluated using the K-10 cross validation in order to select the strict classifier. The strict classifier that used cost 10 scored 0 FN with 9 FP results. The chosen classifier's feature combination was LPF, LIdle and BPush. The strict classifiers for both costs 20 and 30 scored the same results, in terms of the used feature combination and the results in detecting Zeus same version's flows (same FN and FP results). This is explained as both costs provided the same rules and values while building the classifiers. The classifier that scored 0 FN with 5 FP results was chosen as the strict classifier. The chosen classifier's feature combination was MPF, LPF and BPush.

For the second phase, first the classifiers that were selected as the lenient classifier were used to include the untrained version's flows.

It can be noticed from Fig. 1 that a Recall score of 0.667 was obtained for the three costs, reflecting that more than half of Zeus version 2.x flows were included from the dataset. This is because the classifier identified 96 Zeus flows out of 144 flows. The Precision score was 0.676 for cost 10 and 0.686 for costs 20 and 30, which demonstrate that more than half of the flows identified as Zeus flows were actually Zeus for the three costs. This is explained as the FP results were 46, 44 and 44 for costs 10, 20 and 30 respectively. The F-Measure results were 0.671 for cost 10 and 0.676 for costs 20 and 30.

Second, the strict classifiers identified earlier were used, in order to filter out the FP flows that were included in the previous step by the lenient classifiers. All the flows that were predicted as Zeus from the previous step are to be used as an input to these classifiers. Accuracy results are demonstrated in Fig. 1 for the different costs. The Recall result was 0.667 for cost 10 strict classifier, indicating that the classifier did not filter out any Zeus flow that was included by the lenient classifier. On the other hand, the strict classifiers for costs 20 and 30 filtered out 8 Zeus flows that were included by the lenient classifiers, reducing the Recall results to be 0.611. Also, it can be seen that the strict classifier for cost 10 filtered out 42 non-Zeus flows that were classified as Zeus by the lenient classifier, enhancing the Precision result to be 0.960, where costs 20 and 30 filtered out 43 non-Zeus flows that were classified as Zeus by the lenient classifiers, improving the Precision results to be 0.989. Although the Recall results declined, the F-Measure results enhanced for all the costs as the Precision improved significantly, providing 0.787, 0.755 and 0.755 for costs 10,20 and 30 respectively.

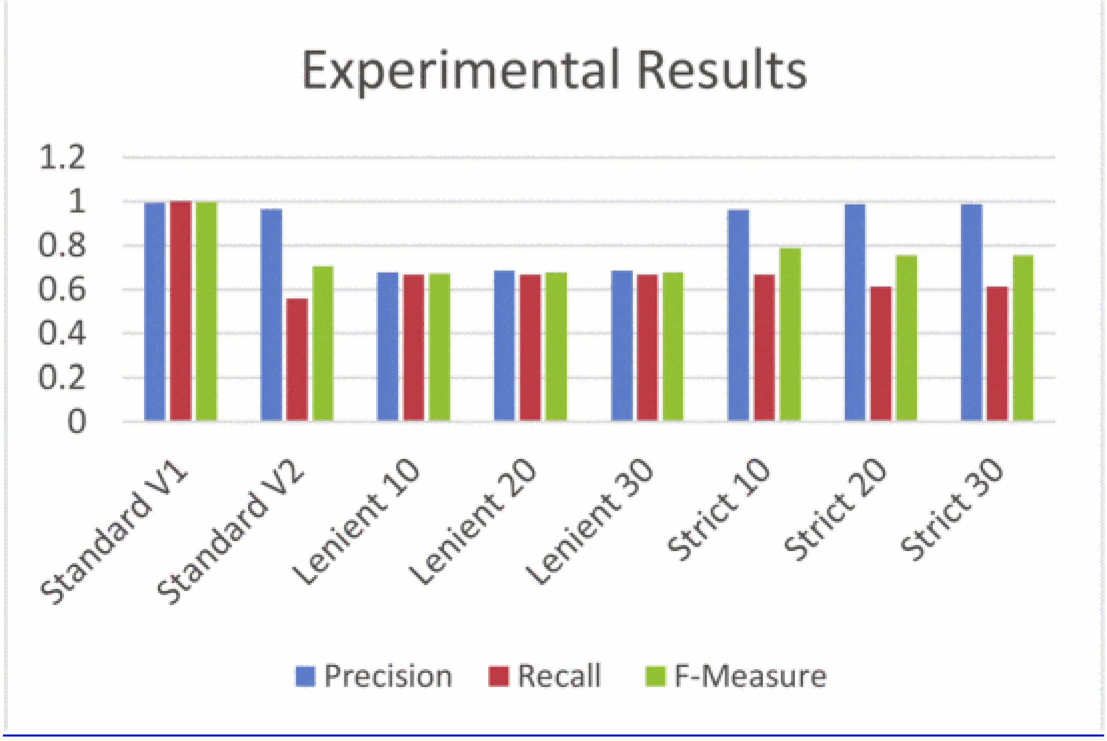


Fig. 1.

Standard and CONIFA detection results

The voting results from the three costs achieved 88 TP, 56 FN and 1 FP results, providing 0.989 Precision, 0.611 Recall and 0.755 F-Measure results. These results indicate that the use of the strict classifier might affect the detection of the untrained version's flows that were included by the lenient classifier. If we use the lenient classifiers by themselves, the voting results would be 96 TP, 48 FN and 44 FP results, providing 0.686 Precision, 0.667 Recall and 0.676 F-Measure results. Even though the results showed a higher FP, they showed a lower FN, which is more important in this case as missing an infected device has a higher impact than wrongly classifying a clean device as being infected.

SECTION VI.

Discussion

This section provides a discussion of the results attained by the standard and CONIFA frameworks in detecting the untrained version's flows of Zeus application for the tested feature selection and learning algorithms.

The standard machine learning framework provided good accuracy results in identifying the untrained version of Zeus malware, utilizing the CFS feature selection algorithm with the C4.5 learning algorithm To address the cause of this behaviour, we investigated the utilized features by the built classifier and observed that C4.5 primarily deployed LPF (Largest Packet Forward) and BPush (Number Push Backward) features. For these two features, we addressed the values attained by the trained and untrained versions of Zeus flows as well as non-Zeus flows.

Fig. 2 demonstrates the distribution of the flow's values against the two features, where the blue colour represents Zeus version 1.x flows, the red colour represents Zeus version 2.x flows and the green colour represents non-Zeus flows.

The first remark that can be conclude from the figure is the validity of our theory, which states that different versions, trained and untrained, for a targeted application share common behaviour and features to each other more compared to non-targeted application. Zeus versions 1.x and 2.x flows are nearer to each other in comparison to non-Zeus flows, so maximising the identification of the trained version by getting low FN and high FP results magnifies the identification of the untrained version. The second remark is that training the classifier on a single version utilizing cost insensitive algorithms and the single feature combination concepts lead to good performance results in detecting the untrained version, although it is less compared to the detection of the trained version. This is justified as the utilized features have some similar values for both versions of Zeus, which maximised the probabilities to identify the untrained version by training the classifier on a single version. Further, the same version detection provided 0 FN and 3 FP results, which maximised the possibility of identifying the trained version, thus the untrained version.

CONIFA framework enhanced the accuracy results in identifying the untrained version of Zeus malware (identifying 96 flows), utilizing the CFS feature selection algorithm with the C4.5 learning algorithm Fig. 3 illustrates the distribution of Zeus versions 1.x and 2.x as well as non-Zeus flows using the feature space for LPF (Largest Packet Forward) and LIdle (Maximum flow Idle) features. These features were primarily used by the lenient classifiers resulted from CONIFA C4.5. Similar to the previous figure observations, both trained and untrained versions of Zeus share some similar values for the selected features. Further, the two versions flow's values are closer to each other compared to non-Zeus flows.

CONIFA utilizes both cost sensitive algorithms and different feature combinations, unlike the standard framework that uses cost insensitive algorithms and a single feature combination. Cost sensitive algorithms provide the ability to adjust the decision boundary while building the classifier to increase the probability to detect the trained version, thus maximising the chances in detecting the untrained version of Zeus malware. Building different classifiers using different feature combinations provides different classifiers with various FN and FP results. This can be seen as the lenient classifier scored 0 and 34, 35 and 36 FN and FP results respectively for the same version, thus enhancing the detection of the untrained version.

SECTION VII.

Conclusion

Botnet is currently the main threat in the internet against individuals and organizations, causing large losses for both parties. Botnet C&C channel traffic identification is a vital task to treat the infected devices, take C&C server down and track down cybercriminals.

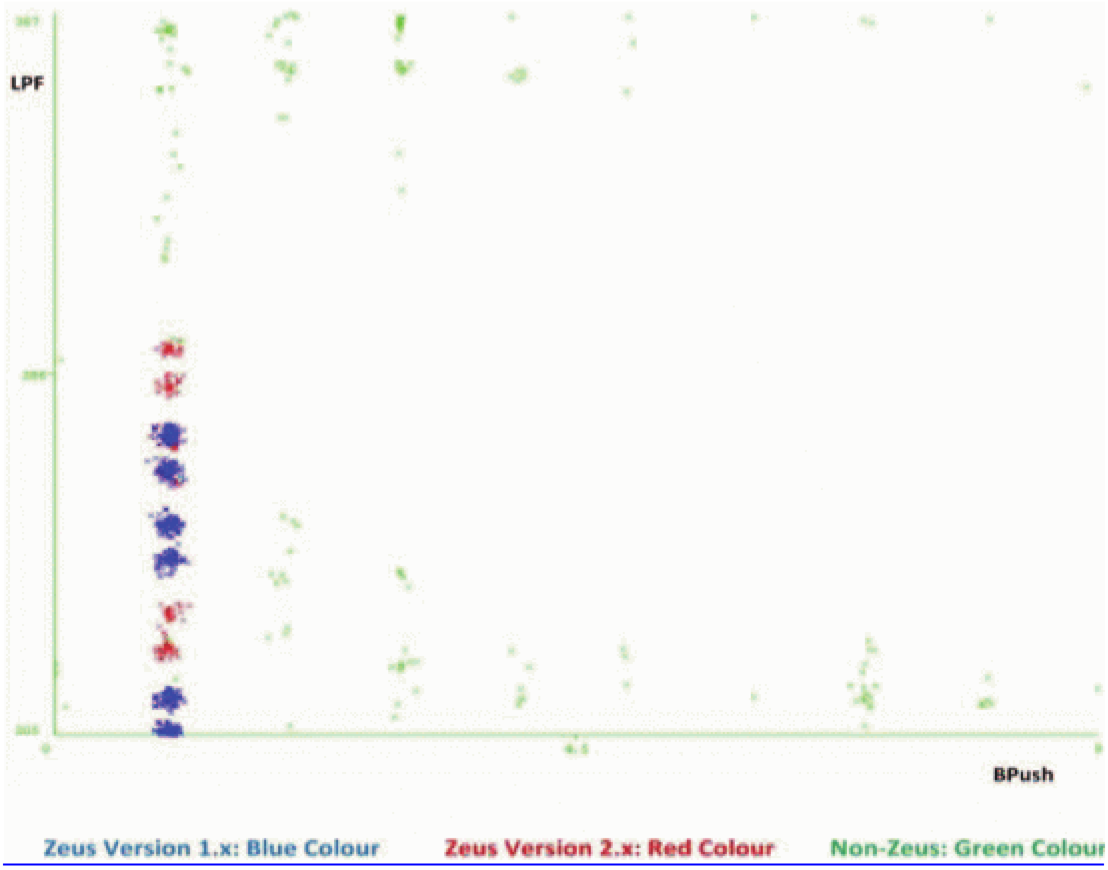


Fig. 2.

Zeus and non-zeus flows distribution for the features used by the standard framework

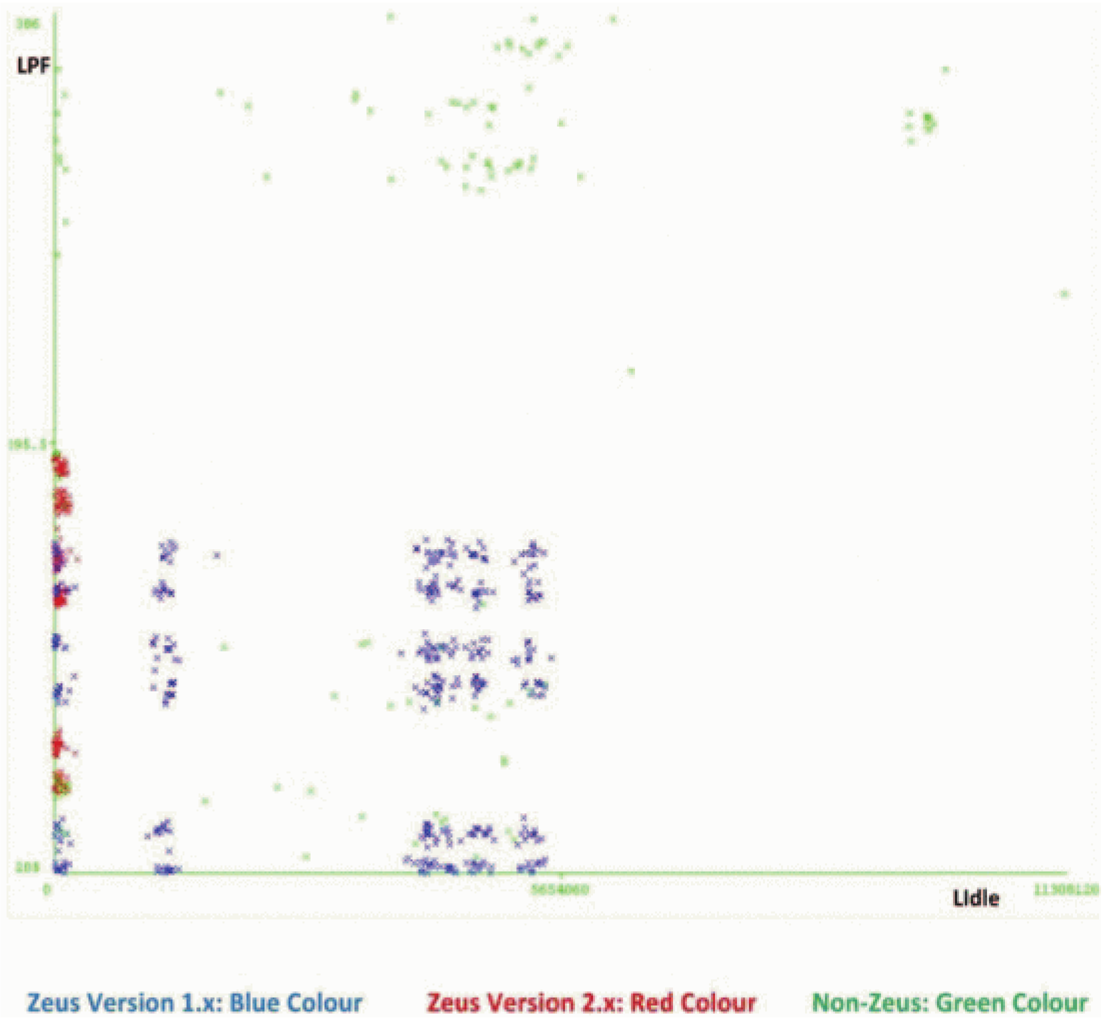


Fig. 3.

Zeus andnon-zeus flows distribution for the features used by the lenient classifier

Researchers proposed approaches as DPI, DNS request behaviour, temporal, correlation and machine learning to detect the C&C channel traffic. The approaches addressed the solutions in terms of online classification capability by monitoring a few packets in a flow, supporting various transport and application protocols as TCP, UDP, HTTP and IRC, avoiding accessing packet's content, relying on a single phase's traffic for the detection process by monitoring only C&C channel traffic, monitoring a single device's network traffic and detecting untrained versions for the targeted application by analysing and building the classifier on a different single version. None of the approaches fulfilled these characteristics in a single solution.

The novel methodology, CONIFA, was used to overcome this gap, by fulfilling all the aforementioned characteristics in a single solution. CONIFA mainly relies on the machine learning approach and aims in filling its gap by the fact that the machine learning approach performance does degrade if the untrained versions have statistical values that are dissimilar from the one used by the built classifier.

CONIFA deploys cost sensitive algorithms and different feature combinations concepts. These two concepts are used to maximise the detection of the trained version, thus increasing the likelihood in detecting the untrained version. The evaluation results showed the efficacy of CONIFA in detecting the untrained version of Zeus botnet by building the classifier on a single version, providing a 0.667

Recall result using only the lenient classifier and a 0.611 Recall result using the strict classifier, compared to a 0.556 Recall result using the standard machine learning framework.

CONIFA proved its validity in detecting zero day versions of a botnet. As a future work, CONIFA performance is to be addressed using more feature selection and machine learning algorithms. Also, other malware's families are to be evaluated using CONIFA.

REFERENCES

[1] H. Binsalleeh, T. Ormerod, A. Boukhtouta, P. Sinha, A. Youssef, M. Debbabi, et al., "On the analysis of the zeus botnet crimeware toolkit," in Privacy Security and Trust (PST), 2010 Eighth Annual International Conference on, 2010, pp. 31-38.

[2] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, et al. , "Detecting P2P botnets through network behavior analysis and machine learning," in Privacy, Security and Trust (PST), 2011 Ninth Annual International Conference on, 2011, pp. 174-180.

[3] M. Stevanovic, K. Revsbech, J. Pedersen, R. Sharp, and C. Jensen, "A Collaborative Approach to Botnet Protection," Multidisciplinary Research and Practice for Information Systems, pp. 624-638, 2012.

[4] D. A. e. al, "Security Intelligence Report," Microsoft2010.

[5] Symantec, " Internet Security Threat Report," 2011.

[6] T. Alpcan and N. Bambos, "Modeling dependencies in security risk management," in Risks and Security of Internet and Systems (CRiSIS), 2009 Fourth International Conference on, 2009, pp. 113-116.

[7] A. Azab, R. Layton, M. Alazab, A. Stranieri, and P. Watters, "Classification of Network Information Flow Analysis (CONIFA) for Detecting New Versions of Network Based Applications," Unpublished
.

[8] J. Wyke, "What is Zeus?," Sophos2011.

[9] P. Coogan. (2010). Zeus, King of the Underground Crimeware Toolkits [Online]. Available: http://www.symantec.com/connect/blogs/zeus-king-underground-crimeware-toolkits

[10] F-Secure, "Threat Report," 2012.

[11] RSA, "RSA 2012 CYBERCRIME TRENDS REPORT," 2012.

[12] Z. B. Celik, R. J. Walls, P. McDaniel, and A. Swami, "Malware traffic detection using tamper resistant features," in Military Communications Conference, MILCOM 2015 - 2015 IEEE, 2015, pp. 330-335.

[13] M. Hogan, F. Liu, A. Sokol, and J. Tong, "Nist cloud computing standards roadmap," NIST Special Publication, vol. 35, 2011.

[14] J. R. Binkley and S. Singh, "An algorithm for anomaly-based botnet detection," presented at the Proceedings of the 2nd conference on Steps to Reducing Unwanted Traffic on the Internet - Volume 2, San Jose, CA, 2006.

[15] C. Hyunsang, L. Hanwoo, L. Heejo, and K. Hyogon, "Botnet Detection by Monitoring Group Activities in DNS Traffic," in 7th IEEE International Conference on Computer and Information Technology. CIT 2007, 2007, pp. 715-720.

[16] F. Giroire, J. Chandrashekar, N. Taft, E. Schooler, and D. Papagiannaki, "Exploiting Temporal Persistence to Detect Covert Botnet Channels," in Recent Advances in Intrusion Detection. vol. 5758, ed: Springer Berlin Heidelberg, 2009, pp. 326-345.

[17] G. Fedynyshyn, M. C. Chuah, and G. Tan, "Detection and classification of different botnet C&C channels," presented at the Proceedings of the 8th international conference on Autonomic and trusted computing, Banff, Canada, 2011.

[18] S. Garasia, D. Rana, and R. Mehta, "HTTP botnet detection using frequent patternset mining," International Journal Of Engineering Science & Advanced Technology, vol. 2, pp. 619-624, 2012.

[19] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," SIGMOD Rec., vol. 22, pp. 207-216, 1993.

[20] G. Gu, R. Perdisci, J. Zhang, and W. Lee, "BotMiner: Clustering Analysis of Network Traffic for Protocol-and Structure-Independent Botnet Detection," in Proceedings of the 17th USENIX Security Symposium, San Jose, CA., 2008, pp. 139-154.

[21] G. Gu, J. Zhang, and W. Lee, "BotSniffer: Detecting botnet command and control channels in network traffic," in Proceedings of the 15th Annual Network and Distributed System Security Symposium. San Diego, CA., 2008.

[22] X. Zang, A. Tangpong, G. Kesidis, and D. J. Miller. (2011, 12 July). Botnet detection through fine flow classification [Online]. Available: http://www.cse.psu.edu/research/publications/tech-reports/2011/CSE-11-001.pdf

[23] G. Kirubavathi Venkatesh and R. Anitha Nadarajan, "HTTP Botnet Detection Using Adaptive Learning Rate Multilayer Feed-Forward Neural Network," in Information Security Theory and Practice. Security, Privacy and Trust in Computing Systems and Ambient Intelligent Ecosystems. vol. 7322, ed: Springer Berlin Heidelberg, 2012, pp. 38-48.

[24] G. Kirubavathi and R. Anitha, "Botnet detection via mining of traffic flow characteristics," Computers & Electrical Engineering, vol. 50, pp. 91-101, 2// 2016.

[25] F. Haddadi, D. Runkel, A. N. Zincir-Heywood, and M. I. Heywood, "On botnet behaviour analysis using GP and C4.5," presented at the Proceedings of the 2014 conference companion on Genetic and evolutionary computation companion, Vancouver, BC, Canada, 2014.

[26] M. Stevanovic and J. M. Pedersen, "An efficient flow-based botnet detection using supervised machine learning," in 2014 International Conference on Computing, Networking and Communications (ICNC), 2014, pp. 797-801.

[27] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, et al., "Botnet detection based on traffic behavior analysis and flow intervals," Computers & Security, vol. 39, Part A, pp. 2-16, 11// 2013.

[28] M. Alazab, "Profiling and classifying the behavior of malicious codes," Journal of Systems and Software, vol. 100, pp. 91-102, 2015.

[29] D. Arndt. (2011, 19 August). Summary of the features output by netmate-flowcalc. [Online]. Available: https://code.google.com/p/netmate-flowcalc/wiki/Features

[30] P. Christen, Data matching: concepts and techniques for record linkage, entity resolution, and duplicate detection: Springer, 2012.

[31] M. Alazab, S. Venkataraman, and P. Watters, " Towards Understanding Malware Behaviour by the Extraction of API Calls" The Second Cybercrime and Trustworthy Computing Workshop (CTC), 2010 Ballarat, VIC, 2010, pp. 52-59.