Computing Conference 2017
18-20 July 2017 | London, UK

Mining malware command and control traces

Peter McLaren, Gordon Russell, Bill Buchanan
School of Computing
Edinburgh Napier University
Edinburgh, UK
p.mclaren@napier.ac.uk, g.russell@napier.ac.uk, b.buchanan@napier.ac.uk

***Abstract*—Detecting botnets and advanced persistent threats is a major challenge for network administrators. An important component of such malware is the command and control channel, which enables the malware to respond to controller commands. The detection of malware command and control channels could help prevent further malicious activity by cyber criminals using the malware. Detection of malware in network traffic is traditionally carried out by identifying specific patterns in packet payloads. Now bot writers encrypt the command and control payloads, making pattern recognition a less effective form of detection. This paper focuses instead on an effective anomaly based detection technique for bot and advanced persistent threats using a data mining approach combined with applied classification algorithms. After additional tuning, the final test on an unseen dataset, false positive rates of 0% with malware detection rates of 100% were achieved on two examined malware threats, with promising results on a number of other threats.**

Keywords—malware; data mining; command and control; anomaly based detection; botnet; advanced persistent threat

I. INTRODUCTION

In the modern internet, cyber criminals are continually looking for ways to carry out malicious activities against users and organisations. Such activities include the theft of data, spamming, and denial of service attacks. Many technologies are available to combat these attacks, with efforts particularly directed at preventing malware from entering the network. However as cyber attackers continually improve their technology, the gap between attack and defence technology means that an aggressive attacker will at some point succeed. Once devices inside a network have been compromised, preventing traffic associated with malware from leaving the network is an essential requirement for organisations in protecting their assets.

Where the malware is a botnet or an advanced persistent threat (APT), the outgoing traffic may contain payloads that launch attacks on other networks, extract confidential data or simply connect with external controllers to receive instructions. Such instructions can direct compromised hosts to initiating network scans, sending spam email, starting distributed denial of service attacks, steal data, or providing means of updating the malware to provide a more effective and resilient platform for the criminals. By detecting and breaking the command and control channel between the client and its controller, the bot can be rendered valueless as it will be unable to respond to commands or report back.

The detection of malware in network traffic is traditionally carried out by recognising specific patterns in packets, defined through the study of known malware activity. However as malware evolves, these patterns may change so that detection is evaded. Thus there is a catch-up period for the detection signatures, during which time malware activity may go undetected. Additionally, malware writers are more frequently encrypting the communications between bot and controller, and so pattern recognition is no longer as effective and newer, more advanced technologies are required. However, as command and control communications are programmed to engage consistently in a terse command and response exchange, it is expected that the resulting network traffic patterns will differ from those resulting from human-initiated exchanges. As command and control communications may demonstrate different characteristics than normal network traffic (e.g. packet size, frequency, and IP header information), it is expected that measurable characteristics exist which differentiate malware from normal network traffic.

This paper investigates multiple features of egress network traffic, including packet and flow characteristics, to enable command and control channels for botnet and APTs to be reliably and efficiently detected. The challenges of detecting botnet and APT command and control communications are investigated and potential solutions to the challenges identified. An algorithm is then proposed using a hybrid data mining system linked with a rule-based network classifier, which is then analysed using a number of traffic flows containing a mix of normal and malware-related communications.

The features of interest in this paper are:

• Analysis of both packet and network flow data. Analysis techniques using purely network flow data will discard packet and packet header data which may identify malware. Models using payloads for classification generally require these to be unencrypted which will be defeated by bots encrypting command and control traffic. This paper proposes an accurate detection scheme which tolerates payload encryption, making use of both flow data and non-payload packet data.

• Use of publicly available, contemporary malware datasets. Studies which use privately obtained datasets are unable to be repeated by other researchers. The use of public datasets enables the analysis on current malware found within this paper to be reproduced.

- Analysis of APT command and control channel traffic using data mining analysis techniques. To our knowledge this is a little explored research area.

II. Background

A. Malware

The APT kill chain has been described as having seven elements: reconnaissance, weaponization, delivery, exploitation, installation, command and control, and actions on objectives [1]. The identical kill chain has subsequently been used to describe bot attacks [2]. In both instances command and control was described by reference to manual external actors, which directed activities on compromised hosts. Although the lifecycles of each bots and APTs differ, it is proposed that this commonality warrants considering the analysis of intrusion detection mechanisms for both APT and botnet command and control channels in the same study. This paper therefore focuses on both botnet and APT detection

A bot is a piece of malware which runs on a compromised host computer and which responds to commands from a remote controller. Once successfully installed on the target by the attacker (perhaps via an unpatched host vulnerability) a command and control channel is established, over which there is a continuous process of interaction between the bot and the bot controller. Typical activities consist of the bot registering with a controller and then receiving and responding to commands from it. This phase generates network traffic crossing network boundaries, and thus should be discoverable by network intrusion detection systems.

After rallying, a bot receives commands from the controller, perhaps instructing the bot to carry out further malicious activities. This phase depends on the persistence of the command and control channel. Detecting and interrupting that channel will limit the damage caused by the attacker. This phase also generates network traffic which crosses network boundaries.

An APT has a similar lifecycle to a bot. The principal differences between the two malware types are in the malware target selection and the commands that the malware is capable of performing. Targets for bots are widespread, whereas APTs target specific organizations which hold information of value to the intruder. An example of a typical APT target is the 2011 attack on RSA's SecurID products [3].

Bots tend to have large command sets so that they can be used for a wide variety of criminal activities. For example, the default version of Agobot has over ninety commands available to a controller [4]. APTs are generally used for specific purposes such as the destruction of hardware used in a nuclear program [5] or for the exfiltration of data [6]. Consequently the command set for an APT will generally be smaller than that for a bot.

Payloads between APT hosts and controllers contain command and response or exfiltrated data traffic, and this is similar to botnet traffic. Thus analysis of botnet command and control (C&C) traffic should be applicable to that of APTs.

If C&C traffic is transmitted unencrypted, such traffic can be detected by searching for typical character sequences in the message [4]. Newer botnets consequently have encrypted payloads [7] making such detection technique ineffective, so this paper investigates the features of malware packet headers or flow characteristics instead of the payloads. An example of a possible distinguishing feature is message length. When the initial network traffic between bots and controllers was analyzed in one study, it was found that C&C message lengths were generally less than 600 and clustering around 10[8]. Message lengths in normal packets, such as HTTP traffic generated by user browsing, are expected to be considerably larger, making it a possible feature of interest. Using an anomaly detection approach is significantly different from the more common signature approach employed by many security systems in use today. However, working out which distinguishing features to use for anomaly detection, while avoiding labelling legitimate traffic as anomalous, requires the development of an intelligent process.

B. Features of anomalous traffic

In taxonomic terms, one approach to feature identification for botnet traffic is that of application-based systems [9]. This includes such techniques as graph theory, machine learning, decision trees, and data mining. Data mining is specifically explored and evaluated in this paper as a tool for identifying appropriate malware traffic features. Even here, there are a wide range of taxonomic opinions as to how to define data mining approaches, such as an approach which splits numerical data into its own group [10]. The taxonomy shown in Figure 1, is adapted from a standard mining textbook [11], and will be used in evaluating a variety of feature identification methodologies, and supporting a range of data types across all groups.

Classification [12] [13], association [14], time series [15] [16] and clustering [17], [8] algorithms have been used previously for data mining command and control traffic. For example, classification was used to determine flow characteristics in an IRC bot [13], association rules were used to identify common characteristics of anomalous data flows [14], time series analysis on network traffic was used to enable the application of change point detection algorithms [16], and clustering was used to determine that message length in botnet traffic was clustered around a specific value [8]. Algorithm selection should be based on a range of factors, such as the number of infected hosts generating traffic and the size of the training dataset. Our work can be distinguished from the above either because it is sufficient four purposes to have a single trace rather than multiple bot traces [13], [14], [8], [17] or because we are mining both packet and flow data [12], [15]. As training datasets may only capture traffic from a single bot or APT, classification was considered to be the best approach to use in this research.

Within the classification approach, there are many different algorithms which could be applied. For the detection of malware command and control traffic, the algorithm should be accurate (so as to avoid false positives [18]), robust (able to deal with noisy data and thus less sensitive to the quality of a mix of training data sets), deal with outliers (where some data elements are significantly different in the data set), and handle non-numeric data (thus avoiding translating non-numeric data from the samples into an artificial numeric mapping). Using a criteria-based analysis of Kotsiantis [19], the Decision Tree class seems to perform best in those categories, although Naïve Bayes is a close second.

Initial experiments with Naïve Bayes highlighted some significant issues, but most importantly is that Naïve Bayes uses a Gaussian probability distribution over a range for continuous variables. This leads to unrealistic results such as ranges including negative packet sizes, so that when using this approach in the experiments classification accuracy was poor. Therefore Naïve Bayes was dismissed as an approach even though earlier studies appear to have applied the algorithm successfully [20].

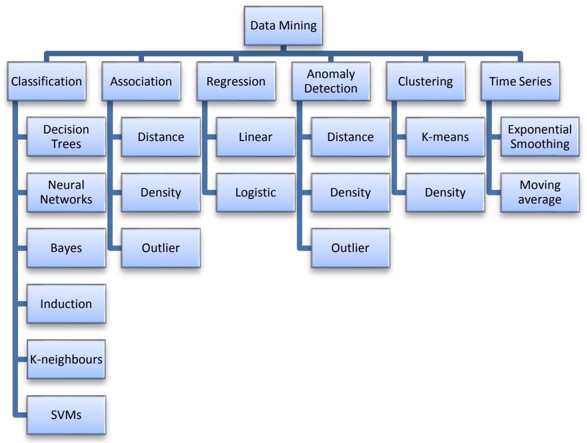

Figure 1 Data Mining Taxonomy

III. Approach

In order to analyse the effectiveness of using data mining to detect C&C traffic, an experiment was constructed which took the form of three phases. In Phase 1, packet captures were obtained or created which contained a mix of normal traffic with the C&C traffic. In Phase 2, this traffic was cleaned by removing derived and redundant fields, labelled as normal or malware, and then mined to reveal features of interest (such as frame length) which could be used to classify the C&C traffic. At this phase some analysis of the results was also undertaken. In Phase 3, the mined feature list was converted to rules, applied to a network classifier tool, and then used to classify traffic captures which were different from the captures used in the mining phase. The resulting performance was then evaluated.

When measuring performance of a network classifier, there are two key measures. Firstly the rate at which good traffic is classified incorrectly as C&C traffic (False Positive Rate or FPR) should be low or preferably 0. A zero FPR would mean no good traffic would ever be labelled incorrectly as bad traffic. Secondly, C&C traffic should be detected effectively, so that as few packets as possible will pass before there is a high confidence that the bad traffic would be detected. The detection rate R_{bad} is the probability of identifying a bad packet as bad, and R_{good} is the probability of identifying a good packet as good.

Given a stream of P good packets and N C&C packets, the number of packets identified as bad packets which truly are bad are true positive TP packets. Equally the number of packets identified as bad which are in fact good are false positive FP packets. Additionally the number of packets which were bad but incorrectly identified as good is TN. Thus:

$$FPR = \frac{FP}{TP + FP} \quad R_{bad} = \frac{TP}{TP + TN}$$

A low FPR means that good packets are rarely identified as bad, and since believing a packet was bad could lead to the packet being processed for additional checks, or simply dropped, or else leading to an alert of some sort, having a low FPR is the goal. A high R_{bad} indicates that it is more likely that bad packets will be identified quickly, with an R_{bad} of 1 indicating that all bad packets are always identified.

A factor which can hinder experimental investigations is ensuring the datasets used for training and testing are both realistic and also cover a range of malware. In the initial experiments a virtualised network and Windows PC environment was created. The traffic on this network was captured while the virtual PC was used to access a range of external websites. This dataset, which contains no malware traffic, could therefore be well understood and investigated should any issues arise with the dataset later. This captured dataset was then mixed with malware PCAP captures obtained from the Contagio website [21]. This website gives access to a wide range of captured botnet and APT traffic streams. The mixed dataset is designated the hybrid dataset and is used for training.

Before the hybrid dataset was mined for features, it was pre-processed in a number of ways. TShark [22] was used to extract the dataset into different files, each focusing on a different protocol (i.e. Ethernet, IP, TCP, DNS, and so on). The files were also cleaned, to remove fields which had the same value for each packet, were inapplicable because of the mixing of the different datasets (e.g. MAC address) or were used in labelling (source IP addresses) or where one field was derivable from a different field (redundant information). This makes the mining process faster, and should increase accuracy [23].

For the network traces used in our research, TCP was the command and control communications protocol. UDP was used, in DNS lookups, for example, but it was not evident in the command and control channel. Although UDP is claimed to be increasingly popular with malware writers [24], this may be largely for bot-to-bot, i.e. peer-to-peer communications, and not necessarily for communications between bot and controller [25]. The absence of UDP in the command and control traffic may also be a consequence of the botnet and APT candidates selected for the current study.

For the experiments, the widely used RapidMiner [26] data miner was employed to perform the mining process in preference to Weka [27], as it was felt that RapidMiner offers the best features, as well as a user-friendly interface, along with a large set of input formats.

IV. EVALUATION

A. *Hybrid Dataset*

The hybrid dataset was filtered and processed. Each resulting training set was mined with RapidMiner, but only the TCP set performed well when each set was considered individually. The results of the TCP mining experiment are shown in Table 1. In this table, the classification rules from the decision tree have been extracted and converted to IF-style rules. The rules are listed in priority order, from most effective to least effective. Although not capturing the complete detail of the decision tree information, it does depict the classification process in a way which could be approximated in a network classifier.

Table 1 TCP Packet-Based mining on hybrid dataset

Malware Name	**Classification results**		
	Ordered Feature List	***FPR***	R_{bad}
Zeus	1. window size > 64376 2. window size <= 64376 and header length > 30 3. window size <= 64376 and header length > 30 and window size scale factor > 1.5	0.00%	100%
Zeus Outbound	1. header length < 30	0.00%	6.67%
Cutwail	1. window size value > 64887 2. window size value <= 64887 AND destination port <= 52	0.00%	63.66%
Zeus Gameover	1. frame length <= 57 2. frame length > 57 and window size value > 64245 3. frame length > 57 and window size value <= 64245 and header length > 30	0.00%	50.78%
Citadel	1. frame length <= 57 and source port > 1055 2. frame length > 57 and delta time > 4.967 and header length < 24	5.56%	56.67%
AlienSpy	1. delta time < 0.065 and frame length < 59 2. delta time > 0.065 and frame length < 72 and FIN flag <= 0.5	5.15%	86.94%
IRCBot	Insufficient TCP packets		
xTreme RAT	1. frame length < 59 and delta time > 0	2.86%	6.26%

Of the eight bots which generated results, five produced a classification technique with a 0% false alarm rate. R_{bad} is generally high, though not generally 100%. This suggests that the packets will be identified eventually, but perhaps not on the first packet. These five rules could be directly mapped into an intrusion detection scheme without significant difficulties. However, other rules performed less well.

The rules which performed poorly were investigated, and the training sets examined. In the case of "Zeus Outbound bot", other features could have been detected which seemed to be statistically significant in the captures, but these had been unfortunately excluded in the pre-processing step. The balance between fast mining and a wide input data set is something which will need to be investigated further.

Of the remainder of the poorer performing rules, these are largely down to small training datasets, where the malware captures were composed of a limited numbers of packets. The quality of the training set is a challenging issue, as there are few captures generally accessible to researchers. However, Table 1 does show that the general approach has excellent possibilities.

An alternative approach to examining packets is to mine on flow information. This is particularly straight-forward with TCP traffic, where the start and end of a packet flow are clearly marked. UDP traffic would be more challenging, but provided the protocol was understood (such as in the case of DNS lookups) then this could be applied to other traffic types. To explore this, the hybrid flow was pre-processed to produce only flow-based summary information of the TCP information, and removing fields which could mislead the classification process with unwanted traits (such as source IP and source port). The results of this experiment are shown in Table 2.

Table 2 TCP Flow-Based mining on hybrid dataset

Malware Name		**Classification results**		
	Flows	***Ordered Feature List***	***FPR***	R_{bad}
Zeus	18	No features found		
Zeus Outbound	9	No features found		
Cutwail	161	1. Destination port <39 and >12	0.00%	46.15%
Zeus Gameover	418	1. IP bytes received >56 and 80 < IP bytes sent <897 2. IP bytes received <=56 and IP bytes sent <= 2242) and IP packets sent <=2)	8.06%	91.20%
Citadel	31	1. IP packets received ==1 2. Destination port <=107 3. IP packets sent >=4	18.18%	100%
AlienSpy	3626	1. 784 < destination port <1112 2. 44 < IP bytes sent <=50 and IP packets received <= 10	0.00%	100%
IRCBot	17	Insufficient data		
xTreme RAT	18	1. IP bytes sent >1043885	50.00%	60%

Table 2 suggests that this approach may offer an additional dimension to the detection of C&C traffic. However, the FPR is often above 0%, which would make this approach error-prone as a technique on its own. One factor which may be an issue is the small sample size of some of the flows. No definitive formula was found in published works that identified a minimum sample size, though it has been suggested that 30 should be the minimum [28]. Thus this may be an important factor when considering the performance of the flow-based analysis.

B. *Mining Rules in Network Classification*

The next phase of the work was to use the decision trees generated in the analysis, and convert them for use in a network flow analysis tool. Bro [29], which was originally written as an intrusion detection tool [30], also works effectively as a flow analysis tool, and furthermore is extensible through the use of its scripting language. A simple mapping process was employed with the packet-based rules, taking the decision tree and writing it directly as Bro rules. Such a simple process did result in some loss of accuracy, but was still sufficient to support the experiment. Other approaches, such as using a weighted average, will be considered as future work.

As part of the experiment, the miner rules for flows were also utilised in Bro. However, rather than a direct translation, a hybrid algorithm was introduced. The hybrid algorithm used the flows data miner decision tree, but combined with the packet-based decision tree, so that if a flow contains a packet which is classified as malware, then the whole flow is also marked as malware.

The performance of the packet-based only Bro rules were compared against the performance obtained from the data miner, and this is shown in the Packets column of Table 3. The table also compares the flow-based decision tree against the hybrid Bro rules. The performance of the hybrid algorithm is noticeably higher than all other measures, with a FPR of generally 0% and an R_{bad} of generally 100% (although Zeus Gameover bot continues to perform slightly worse than the rest).

Table 3 Bro rules applied to Hybrid Dataset

Malware Name	*% Performance (packets)*				*% Performance (stream)*			
	Miner		*Bro*		*Miner*		*Bro*	
	FPR	R_{bad}	FPR	R_{bad}	FPR	R_{bad}	FPR	R_{bad}
Zeus	0.0	100	8.2	38.3	-	-	0.0	100
Zeus Outbound	Unreliable packet and flow feature identification							
Cutwail	0.0	63.7	0.2	63.7	0.0	46.1	0.0	100
Zeus Gameover	0.0	50.8	0.1	50.1	8.06	91.2	0.0	72.7
Citadel	Unreliable packet and flow feature identification							
AlienSpy	5.2	86.9	0.7	4.9	0.0	100	0.0	100
IRCBot								
xTreme RAT	Unreliable packet and flow feature identification							

C. *Unseen Dataset*

In order to verify the effectiveness of the hybrid flow-based Bro rules, the Bro scripts were run on a packet segment taken from the ISOT Botnet packet capture file [31], which had not been used in any part of the data mining training process. The results, shown in Table 4, are quite poor, and significantly lower than expected. It should be noted that small sample sizes of some of the malware in this capture file made some classifications impractical, and so obtaining better data samples is a goal for the future. However, the datasets are more than adequate to allow the poor performance to be explored further.

Table 4 Bro Rules on independent data set:1st pass

Malware Name	**Packets**	**Flow**
	FPR	**FPR**
Zeus	99.4%	96.7%
Zeus Outbound	Unreliable flow feature identification	
Cutwail	74.33%	87.71%
Zeus Gameover	88.12%	92.64%
Citadel	Unreliable packet and flow feature identification	
AlienSpy	0.67%	99.02%
IRCBot	No features identified	
xTremeRAT	Unreliable packet and flow feature identification	

In examining the ISOT capture file, one of the most significant aspects noted is that "window size", which plays a significant role as a primary rule in classifying many of the bots, now appears to be a poor discriminator. The key issue here is that some features identified at the mining stage may prove to be unsatisfactory discriminators in some environments, so extensive datasets are essential. This may be caused by identifying features particular to the dataset-capturing environment, rather than general characteristics. Implementing a classification algorithm with more tolerance may also result in better performance. For the purposes of the investigation, the window size rules were simply deleted from the classification script. A similar issue arose for AlienSpy, as the feature related to its destination port also was a poor discriminator in the ISOT file. Again the rule was simply deleted. The experiment was then performed again on the ISOT capture using the edited Bro rules, and this is shown in Table 5.

Table 5 Bro Rules on independent data set: 2nd pass

Malware Name	**Packets**	**Flow**
	FPR	**FPR**
Zeus	99.4%	96.7%
Zeus Outbound	Unreliable flow feature identification	
Cutwail	74.33%	87.71%
Zeus Gameover	88.12%	92.64%
Citadel	Unreliable packet and flow feature identification	
AlienSpy	0.67%	99.02%
IRCBot	No features identified	
xTremeRAT	Unreliable packet and flow feature identification	

As can be seen in the analysis, the resulting hybrid flow-based Bro classification remains highly effective for both Zeus and AlienSpy. Zeus Gameover performance is identical to the hybrid dataset results. Only the Cutwail flows remains poor with an unacceptably high FPR.

In a further analysis of Cutwail many possibilities for improving detection methods are highlighted. It is known from the Contagio captures that Cutwail should make a number of

DNS lookups (although a significant percentage of DNS responses appear to be truncated). Data mining experiments on the hybrid dataset DNS lookups shows that Cutwail has a number of DNS-protocol features which form good feature classifiers, and so this data could be merged with the Bro rules in a system which was able to consider multiple protocols in order to make decisions. Additionally, mining the HTTP protocol for HTTP features also highlights peculiarities in how Cutwail makes HTTP requests, which again could be used in a multi-protocol approach. By using a multi-protocol approach it is likely that even hard to classify traffic, such as Cutwail, would become easy to classify. Such an approach should also make other bot traffic classification more reliable. This is considered as the next stage of this research.

V. CONCLUSION

The overall aim of this paper is to demonstrate the effective classification of malware packets so that command and control channels could be reliably and efficiently detected. A hybrid ruleset derived from a data miner decision tree was shown to be highly effective at targeting the TCP-related C&C traffic, so long as care is taken in interpreting the resultant rules. Applying the algorithms to the unseen dataset yielded false positive rates of 0% and a malware detection rate of 100% for the Zeus bot and the AlienSpy APT. Reasonable rates were also obtained for the Zeus Gameover bot results and a promising approach was found for the less well performing algorithm for Cutwail. The high performance of this approach appears to be very promising, and it is expected that the classification performance can be significantly improved by combining the TCP-based rules with other bot-related protocol information (such as DNS).

REFERENCES

[1] E. M. Hutchins, M. J. Cloppert, and R. M. Amin, ‘Intelligence-driven computer network defense informed by analysis of adversary campaigns and intrusion kill chains’, *Lead. Issues Inf. Warf. Secur. Res.*, vol. 1, p. 80, 2011.

[2] B. Harris, E. Konikoff, and P. Petersen, ‘Breaking the DDoS attack chain’, *Inst. Softw. Res.*, 2013.

[3] Warwick Ashford, ‘RSA hit by advanced persistent threat attacks’, *Computer Weekly*, 18-Mar-2011.

[4] Honeynet2011, ‘Appendix A: Botnet Commands - Which commands the bots understand’. [Online]. Available: https://www.honeynet.org/book/export/html/55.

[5] Noa Bar-Yosef, ‘When the Advanced Persistent Threat (APT) Meets Industrialization’, 16-Nov-2010.

[6] Security Lancaster, ‘Detecting and Preventing Data Exfiltration’, Lancaster University.

[7] A. Zand, G. Vigna, X. Yan, and C. Kruegel, ‘Extracting probable command and control signatures for detecting botnets’, 2014, pp. 1657–1662.

[8] C. J. Dietrich, C. Rossow, and N. Pohlmann, ‘CoCoSpot: Clustering and recognizing botnet command and control channels using traffic analysis’, *Comput. Netw.*, vol. 57, no. 2, pp. 475–486, Feb. 2013.

[9] H. R. Zeidanloo, M. J. Z. Shooshtari, P. V. Amoli, M. Safari, and M. Zamani, ‘A taxonomy of botnet detection techniques’, in *Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on*, 2010, vol. 2, pp. 158–162.

[10] I. H. Witten, E. Frank, and M. A. Hall, *Data mining: practical machine learning tools and techniques*, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.

[11] V. Kotu, *Predictive analytics and data mining: concepts and practice with rapidminer*, 1st edition. Waltham, MA: Elsevier, 2014.

[12] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, ‘Botnet detection based on traffic behavior analysis and flow intervals’, *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.

[13] W. T. Strayer, D. Lapsely, R. Walsh, and C. Livadas, ‘Botnet detection based on network behavior’, in *Botnet Detection*, Springer, 2008, pp. 1–24.

[14] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, ‘Anomaly Extraction in Backbone Networks Using Association Rules’, *IEEEACM Trans. Netw.*, vol. 20, no. 6, pp. 1788–1799, Dec. 2012.

[15] L. Bilge, D. Balzarotti, W. Robertson, E. Kirda, and C. Kruegel, ‘Disclosure: detecting botnet command and control servers through large-scale netflow analysis’, in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012, pp. 129–138.

[16] P. Wurzinger, L. Bilge, T. Holz, J. Goebel, C. Kruegel, and E. Kirda, ‘Automatically generating models for botnet detection’, in *Computer Security–ESORICS 2009*, Springer, 2009, pp. 232–249.

[17] H. Zhang and C. Papadopoulos, ‘BotTalker: Generating encrypted, customizable C&C traces’, in *Technologies for Homeland Security (HST), 2015 IEEE International Symposium on*, 2015, pp. 1–6.

[18] J. Han and M. Kamber, *Data mining: concepts and techniques*, 2nd ed. Amsterdam ; Boston : San Francisco, CA: Elsevier ; Morgan Kaufmann, 2006.

[19] S. B. Kotsiantis, I. Zaharakis, and P. Pintelas, *Supervised machine learning: A review of classification techniques*. 2007.

[20] W. T. Strayer, R. Walsh, C. Livadas, and D. Lapsley, ‘Detecting botnets with tight command and control’, in *Proceedings. 2006 31st IEEE Conference on Local Computer Networks*, 2006, pp. 195–202.

[21] M. Parkour, ‘Contagio’, 2015. [Online]. Available: http://contagiodump.blogspot.co.uk/.

[22] Wireshark, ‘TShark’. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.htm.

[23] A. Janecek, W. N. Gansterer, M. Demel, and G. Ecker, ‘On the Relationship Between Feature Selection and Classification Accuracy.’, in *FSDM*, 2008, pp. 90–105.

[24] Paloalto Networks, ‘Application Usage and Threat Report’, 2014.

[25] Andrea Lelli, ‘Zeusbot/Spyeye P2P Updated, Fortifying the Botnet’, 21-Feb-2012. .

[26] RapidMiner, ‘RapidMiner’. [Online]. Available: https://rapidminer.com/.

[27] Machine Learning Group at the University of Waikato, ‘Weka 3: Data Mining Software in Java’. [Online]. Available: http://www.cs.waikato.ac.nz/ml/weka/.

[28] H. Sug, ‘An effective sampling method for decision trees considering comprehensibility and accuracy’, *WSEAS Trans. Comput.*, vol. 8, no. 4, pp. 631–640, 2009.

[29] The Bro Project, ‘The Bro Network Security Monitor’. [Online]. Available: https://www.bro.org/index.html.

[30] V. Paxson, ‘Bro: a system for detecting network intruders in real-time’, *Comput. Netw.*, vol. 31, no. 23, pp. 2435–2463, 1999.

[31] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, ‘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.