Malicious Automatically Generated Domain Name Detection Using Stateful-SBB

Fariba Haddadi[1], H. Gunes Kayacik[2], A. Nur Zincir-Heywood[1], and Malcolm I. Heywood[1]

[1] Computer Science, Dalhousie University, Halifax, NS, Canada
{haddadi,zincir,mheywood}@cs.dal.ca
[2] Glasgow Caledonian University, Scotland, UK
gunes.kayacik@gcu.ac.uk

Abstract. This work investigates the detection of Botnet Command and Control (C&C) activity by monitoring Domain Name System (DNS) traffic. Detection signatures are automatically generated using evolutionary computation technique based on Stateful-SBB. The evaluation performed shows that the proposed system can work on raw variable length domain name strings with very high accuracy.

Keywords: Security, Botnet detection, Evolutionary computation, Data mining.

1 Introduction

In the world of fast growing Internet and online activities which almost everyone has something to share and benefit from, having a secure infrastructure is the primary need to protect users' identity and information. Due to the high reported botnet infection rate and its wide range of distributed illegal activities, botnets- among various types of malwares– are one of the main threats against the cyber security [1].

Every year new reports are published indicating that the number of botnet victims is increasing. In 2010, Damballa Inc. published a paper on the top 10 active botnets indicating that botnet infection rate is rapidly increasing by the average growth of 8% per week [2]. McAfee thread reports also confirm that this growth continues into 2012 [3]. These reports also indicate that new powerful botnets enter the Internet realm every year. Moreover, in response to improvements in detection mechanisms, botnets update themselves as well. From the security perspective, these observations indicate that knowing about a botnet mechanism and detecting it would not always be enough in a condition that the master has the opportunity to upgrade or even change its mechanism completely. In this situation, the botnet monitoring activity should be continuous and the botnet detection mechanism should also upgrade itself by the patterns learned through the monitoring process. In other words, this is an arms race and therefore, automating the detection mechanisms as much as possible will give the much needed headway to the defender side. Thus, in this research, we explore how far we can push to automatically generate signatures based on minimum a priori information in order to adapt to the changes in the botnet upgrades.

Monitoring network traffic at DNS level provides a suitable solution to mitigating botnet attacks because, in addition to its many legitimate uses, DNS can also be used by botnets to manage their infrastructure. In a typical botnet for example, the infected computer may locate the C&C server by querying a list of domain names, which are supplied at the time of infection or after. C&C server will instruct the infected host to engage in malicious activities, such as data ex-filtration, denial of service attacks or serving spam, without user's knowledge. The list of domain names provided to the victim host is large enough so that it cannot be blacklisted manually or at firewall level. Thus, to create a long list of domain names, attackers usually generate the list algorithmically. Generated domains exhibit structural and syntactical anomalies compared to regular domain names. It is therefore possible to detect botnet C&C activity by monitoring high volume access to unusually structured domain names.

To detect these anomalies, we employ an evolutionary computation technique based on SBB [4]. Our proposed system employ a modified version of SBB, hereafter called Stateful-SBB (Stateful Symbiotic Bid-Based Genetic Programming). Where most classification algorithms require features to be defined a priori (need behaviour analysis on botnets conducted by human resources), Stateful-SBB works on the raw domain name strings (no a priori information) and achieves comparable detection rates without requiring a predefined feature set. Avoiding such a requirement is the most important contribution of this paper since this enables the approach to adapt to the changes in the botnet upgrades. The remainder of the paper is organized as follows. Section 2 details Botnet topologies, detection methods and related works in this field. Our methodology and the proposed system are discussed in Section 3. Results are provided in Section 4 and conclusions are drawn in Section 5.

2 Related Work

In this section we will give an overview of how botnets work and the existing detection mechanisms in the literature.

2.1 Botnets: How They Work

A bot program is a self-propagating malware that infects vulnerable hosts known as bots (zombies) and is designed to perform a task after being triggered. The infected bots network is referred as botnet, which is under the remote control of a master called botmaster. Usually bots receive commands from the master through a C&C communication channel and carry out malicious tasks such as Distributed Denial of Service (DDoS), spamming, phishing and identity theft attacks [1] [5].

Unlike the earlier botnets that had a list of exploits to launch on targets and all the commands were set at the time of infection, today a typical advanced bot uses multiple phases to create and maintain a botnet including: initial infection, secondary injection, connection, malicious C&C, update and finally maintenance [1] [5]. In the first phase, attacker infects the victim using several exploitation techniques to find its existing vulnerabilities. Once the target got infected, in the second phase, the shell-code is executed on the victim machine to fetch the image of the bot binary which then installs itself on the machine. At this time, the host is completely converted to a zombie and malicious tasks can run automatically on the host. In the connection phase, the bot binary establishes the C&C channel to be used by the master to send the commands to its bot army (botnet). Finally, when the master needs to update the bots for several reasons such as avoiding an antivirus, changing the C&C server setting, or adding a new functionality, the update and maintenance phase is entered.

It is believed that until 2003, most of the botnets were using centralized topology, utilizing IRC Protocol [6] [7]. Since 2003, not only botnets have started to use several protocols such as HTTP and DNS as well as the decentralized topology, but also they have started to employ fluxing methods to avoid detection. Fluxing is a technique used to move the communication between the victims and the C&C server from domain to domain using the DNS protocol [8]. Therefore, since 2004 DNS is used in botnets to add mobility and to remove the single point of failure [7].

2.2 Botnet Detection

Mainly, there are two approaches for botnet detection [1]. The first approach is based on honeynets. Honeynet-based techniques are mostly useful to realize botnets characteristics and technology but not necessarily detection.

The second approach is based on network traffic monitoring and analysis which are typically classified as: Anomaly-based, or Signature-based. Anomaly-based methods rely on finding network anomalies and unusual behaviors such as high volume of network traffic, which could be the outcome of botnet presence in the network. However correctly modeling the network normal behavior is challenging. On the other hand, signature-based methods create signatures to be used for detection purposes. Necessity of prior knowledge of botnets and their behavior make the detection systems of this kind vulnerable to unknown (new) botnets.

There are several detection mechanisms of the second approach that specifically focus on identifying malicious domain names, which are used by DNS-based Botnets. E. Stalmans et al. developed a system to detect fast-flux domains using DNS queries [9]. Analyzing the DNS query responses, two groups of features were extracted to identify legitimate and malicious queries: DNS and Textual features. Given the extracted features, they employed C5.0 and Bayesian classifiers to identify fast-flux queries. S. Yadav et al. proposed a system to detect malicious automatically built domain names [10]. They used several methods and features to group the DNS queries. Then for each group, metrics such as the Jaccard index were computed to differentiate the domain names. J. Ma et al. employed supervised learning techniques (Naive-Bayes, SVM and Logistic Regression) to detect malicious websites from suspicious URLs [11]. To characterize the URLs, two categories of feature were used: lexical and host-based features. M. Antonakakis et al. presented a dynamic reputation system, Notos [12]. Using DNS query data and analyzing zone and network features of domains, Notos builds models of legitimate and malicious domain names.

In this work, our goal is to explore the application of evolutionary techniques in order to automatically generate signatures to detect botnets based on monitoring and analyzing domain names, specifically the ones that are used by botnets for domain fluxing. To the best of our knowledge, all of the works in the literature employ specific pre-defined features of domain names, however we aim to avoid this by only considering the domain names string sequence.

3 Methodology

Network security administrators proposed different approaches to deal with botnets that apply domain fluxing techniques. Some have used black lists to filter out the known C&C servers' domain names or pre-register the probable domains. Others used anomaly and signature-based detection methods. However, all require some type of knowledge on the Botnet domains or Domain Generation Algorithms (DGAs) to be able to generate the exact same domain lists as the botnets. This is a very costly (resources and time) process and also needs to be repeated each time a new DGA is injected. To this end, we believe that a light weight malicious domain name detector can go a long way. Thus, we propose a detection system just based on the automatic analysis of domain name records without requiring any a priori feature sets.

Indeed, one challenge is that automatically generated domain names are also used for legitimate background communications such as software updates and load balancing. Moreover, various well-known websites such as Google and Facebook also use this type of domains. Therefore, the first step to detect the botnet malicious domains is to differentiate legitimate automatically generated domain names from malicious ones. In this case, we propose a new SBB-based classifier system, Stateful-SBB, which works on domain name record strings using no a priori knowledge. We compare our proposed system against original SBB, C4.5, AdaBoost and Naive-Bayes based classifiers, where all require input based on a priori knowledge.

3.1 Learning Algorithms Employed

Naive-Bayes. A Naive-Bayes classifier is a simple probabilistic classifier based on the Bayes theorem, which assumes that the presence of an attribute in a given class is independent of other attributes. The classifier uses the method of maximum likelihood (probability) for parameter estimation. Given a training set (X,Y) where for each sample (x,y), x is an n-dimensional vector and y is the class label out of k number of classes, $C_1, C_2 \ldots C_k$, the classifier predicts that the sample belongs to the class C_i having the highest posteriori, conditioned on x $(P(C_i|x) > P(C_j|x)$ for $1 \leq j \leq k, j \neq i$ $)$. A more detailed explanation of the algorithm can be found in [13].

C4.5. C4.5 is a well-known decision tree-based learning algorithm, which uses the training data to create a tree structure and then classifies the new samples of the test data using the trained tree model. It employs a normalized information gain criterion to select attributes from a given set to determine the splitting point. In this process, the attribute with the highest information gain value is chosen to be the best point. A decision node is then generated based on the best point. The training process recursively continues on the sub-lists obtained until all of the data samples associated to the leaf nodes are of the same class or the classifier runs out of training samples. A more detailed explanation of the algorithm can be found in [13].

AdaBoost. Machine learning techniques' goal is to generate a rule that can predict the new test samples with a high accuracy. Creating a highly accurate rule is a difficult task but on the other hand, generating a set of rough rules of thumb with moderate accuracy is not that hard. Based on this observation, boosting method starts with finding the rules of thumb called weak learner. Given the training set, AdaBoost calls the weak learning algorithm repeatedly, each time feeding it with a different distribution over the training data. Each call generates a weak classifier. At the end, the algorithm combines the classifiers to a single one that is much more accurate than any of them. A more detailed explanation of the algorithm can be found in [13].

SBB. SBB is a form of genetic programming based learning algorithm. It has a team-based framework in which a group of learners are employed to solve a problem. The algorithm consists of three populations: the point population, the team population and the learner (symbiont) population, Fig. 3. The learner population declares a set of symbionts whereas the team population declares learner teams and finally the point population denotes indexes to subsets of exemplars from the training data. Individuals in the learner population take the form of bid-based genetic programs or a representation consisting of program and (scalar) action. Thus, when evaluating a team, each learner program is executed, but only the learner with maximum output (bid) suggests its action (class label). The process repeats for each exemplar in the point population, and again for all teams. There are three important characteristics of learners in case of bidding. First, each learner bids on the point separately but only the learner action with the highest bid is returned as the team action. Second, using the data set with a fixed number of features for exemplars, the learners bid on each point based on the whole feature set. Finally, each learner resets its registers before bidding on the next point. A more detailed explanation of the algorithm can be found in [4].

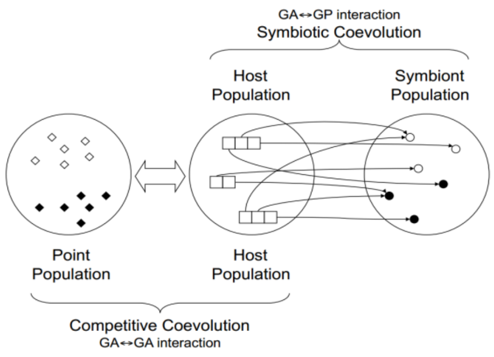

Fig. 1. SBB team-based mechanism [4]

Stateful-SBB. As discussed earlier, identifying the correct set of attributes, which properly represent the domain name characteristics, is challenging especially given that DGAs are moving targets. Thus, to this end, we explored using the raw domain names as an option for detection purposes. Thus, we designed and developed the Stateful- SBB to classify the malicious vs. non-malicious domains by only using the raw domain names. In other words, we explored how far we can push the classification performance without any a priori knowledge about the characteristics of the domain names, i.e. without any lexical features or packet level features.

Learners in the original SBB classifier bid based on all attributes of points (domain names) similar to the aforementioned classifiers. However, given a data set of variable length domains, neither original SBB nor the aforementioned classifiers can be used. Therefore, we change the SBB interface to bid based on each character of a domain name. The new model keeps the state information for each exemplar, hence we call it the Stateful-SBB. Figure 2. summarizes the team-learner interaction mechanism in the Stateful-SBB. Data set exemplars in the new layout are the variable length domain names. Features are the ASCII codes of the domain names' characters. A team receives a complete domain name but it passes the domain name to its team of learners character by character. Each learner then provides a bid per character as opposed to per exemplar. The learners' action that outbids the others is assigned as the team output for that specific character. Domain characters are related to each other and are not independent. To achieve the correlation of characters reflected in the bidding process, learners reset their registers only at the beginning of each specific domain name, not for every bid process on every character in a domain. At the end of each domain name (when all the learners bid on the entire domain name characters), a team will have a sequence of the best learners' actions as the team output sequence. Finally, the team will decide on its final action for that specific domain name. Different policies can be used for the final action selection of a team, which is discussed in the evaluation section.

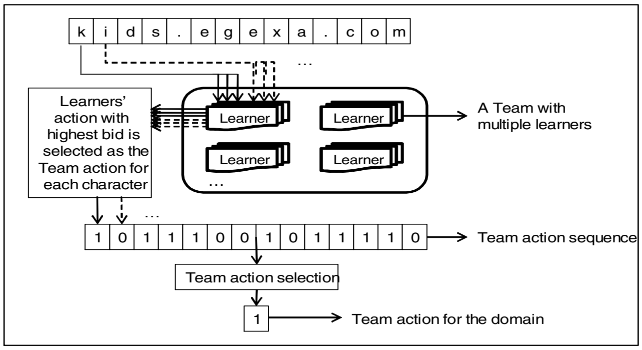
k i d s . e g e x a . c o m
...
Learners' action with highest bid is selected as the Team action for each character
Learner
Learner
Learner
Learner
...
A Team with multiple learners
1 0 1 1 1 0 0 1 0 1 1 1 1 0
Team action sequence
Team action selection
1
Team action for the domain

Fig. 2. Stateful-SBB mechanism

4 Evaluations and Results

In this work, the data set employed is collected from various resources including the publicly available botnet C&C domain lists such as Amada [14] and ZeuS [15]. Additionally, most frequently requested domains from the Alexa list [16] are used as the legitimate domain names. These include known C&C activity as well as social network sites such as Facebook backend and antivirus upload. Thus, the string format resulting data set goes beyond just the C&C traffic and includes other legitimate traffic observable at DNS level. Table 1 details the DNS data set employed in this work. "Class 0" represents normal automatically generated domain names and "class 1" represents the malicious automatically generated ones.

Other than the Stateful-SBB classifier, other classifiers require the data set to be feature-based (a priori knowledge) for training and testing. Thus, we employed a heuristics-based feature extraction on the components of a given domain name to compare against our proposed system. Each domain name has three components: (i) top level domain (TLD), (ii) core domain, and (iii) sub-domain. For example, in mail.google.com, com is the TLD, google is the core domain, and mail is the sub-domain. Given that the TLD names are distinctive and fixed, we only use the other two components (core domain and sub-domain) in feature extraction. Thus, in this work, for each domain name, a set of 17 features are extracted, Table 2. The first 14 features are based on the sub-domain and the last 3 are based on the core-domain component. Overall, the features aim to highlight the structural anomalies in the domain names (as seen in the literature). In other words, domain names that are not likely to be typed by a person. To detach the top-level domain and to extract the feature #12, we employ the Mozilla suffix list [17].

Table 1. Summary of the DNS data set employed

Data set	Data set total Num. of Samples			206
	"Class 0" total Num. of Samples			123
	"Class 1" total Num. of Samples			163
	Training		**Testing**	
	Class0	**Class1**	**Class0**	**Class1**
	90	116	33	47

Table 2. Feature set definition

No.	Features
1	Domain starts with "www"
2	Total sub-domain length: number of characters in all sub-domains (minus the dots)
3	Number of sub-domains: number of sub-domain blocks.
4	Maximum sub-domain length: the largest sub-domain block length
5	10+ character sub-domain count: number of sub-domains longer than 10 chars
6	1 character sub-domain count: number of sub-domains with one char
7	Contains IP: A Boolean flag. If there exists four sub-domain blocks between 0-255, following each other.
8	Alphabetic ratio: Num. of alphabetic character in all sub-domains divided by character count
9	Hexadecimal ratio: Num. of hexadecimal digits (A-F,a-f, 0-9) divided by character count
10	Standard deviation of sub-domain lengths
11	Non-alphanumeric ratio: Number of non-alphanumeric characters
12	Contains imbedded TLD, if the sub-domains contain any items in the Mozilla suffix list
13	Contains imbedded file extension
14	Number of alphabetic to non-alphabetic and vs. transitions
15	Core-domain length
16	Core-domain alphabetic character ratio
17	Core-domain alpha to non-alpha and vs. transition count

We trained each classifier (NB, C4.5, AdaBoost, SBB and Stateful-SBB) on the training data set to identify maliciously generated domain names from the non-malicious ones. Then, the trained models are tested on the test data set. To this end, we divided the dataset into two parts (training and testing) based on: (i) An almost 30-70% breakdown for test and training, respectively; and (ii) keeping enough samples of each class in both of the data sets. It should be noted here that default parameters in WEKA [18] are used for Naive Bayesian, AdaBoost and C4.5 (pruned) classifiers, whereas parameters given in [19] are used for both the original and the Stateful-SBB.

Table 3 presents the results of these experiments. As the results show, the best performers are the C4.5 classifier and the Stateful-SBB classifier. These results show that it is possible to identify the maliciously generated domain names with a high detection rate and a low false positive rate even without a priori knowledge, i.e. without a specific feature set extracted from lexical attributes of a domain name.

It should be noted here that we employed exactly the same data set for all the classifiers. The only difference is for the classifiers other than Stateful-SBB, we represented each record of the data set using the 17 features given in Table 2. However, in the case of Stateful-SBB, we represented the data set in its ASCII code to the classifier. As discussed earlier, the team final action of the Stateful-SBB should be chosen from its learners' output sequence, which is constructed during the bidding process using the domain characters. Given that the domain names are a composition of related characters in a meaningful order, there are some important questions that need to be answered: Is it necessary to use all the domain name characters in the learning process to have a relatively good output label? Should the combination of all actions in the sequence be considered or just the last one? We run several experiments (7 different approaches to the Stateful-SBB and 20 runs for each approach) and evaluated the proposed system on different action selection methods to answer these questions. Because of page limitations, we are not able to present all of these experiments. However, our experimental results showed that the best performances were achieved by the "Last-best" and the "Most-freq" team action selection methods. The "Last-best" method assumes that the class label for the domain name is that returned at the last character. As the learners would not reset their registers in the bidding process of a domain, the last action of the sequence is somehow affected by all the actions in the sequence, where all the domain characters are considered. As the "Last-best" heuristic team might not always reflect all the best actions of bidding process, "Most-freq" method chooses the most frequent action of a team action sequence to be the team final decision. So these versions of the Stateful-SBB are chosen for the evaluation of the proposed system against the other classifiers. Given the results of the Stateful-SBB with two proposed settings, it can be concluded that our proposed Stateful-SBB using the "Last-best" action selection procedure is slightly better than the others. The Results are shown in Table 3.

However to be more precise, we run a T-Test (between the pruned C4.5 classifier and the Stateful-SBB) on 20 different experiments of each solution. The T-Test results indicate that there is not a statistically significant difference between the pruned C4.5 based classifier, which requires a priori known feature set, and the Stateful-SBB, which does not require such a priori information. In summary, these results show that the proposed system employing the new Stateful-SBB has a high classification rate with zero false negative rate for "class 0" (non-malicious domain names). Given that misclassifying legitimate domain names as malicious ones (which ends in a blocking action) can interfere with a genuine website activity, the performance of Stateful-SBB is very promising and shows that it could be deployed in real world environments. Moreover, the most critical phase for all the classifiers using a set of pre-defined features (Naive-Bayes, AdaBoost, C4.5 and SBB) is the feature extraction. In this phase, identifying a correct set of features (a priori information) that can properly represent the domain names is crucial. However, even if a valuable feature set can be identified, since botmasters change their DGAs frequently to evade the detection systems, a feature set that works before, could be stale when the DGA changes! On the other hand, the Stateful-SBB can be applied directly to the domain name strings (using their ASCII codes only) without requiring any feature extraction, in other words without any a priori knowledge. Having said this, the Stateful-SBB training time (computation time) is longer than the others. However, because the training can be performed offline, we believe that the benefits of the proposed system can be considered as a major improvement in this field.

Table 3. Evaluation results on the test data set

		Naive Bayes	Pruned C4.5	AdaBoost	SBB	"Most-freq" Stateful-SBB	"Last-best" Stateful-SBB
Classification Rate		0.95	0.96	0.95	0.95	0.98	0.99
Class 0	Precision	0.94	0.92	0.91	0.94	1	1
	Recall	0.94	1	0.97	0.97	0.94	0.97
	F-Measure	0.94	0.96	0.94	0.95	0.94	0.97
Class 1	Precision	0.96	1	0.98	0.97	0.94	0.97
	Recall	0.96	0.94	0.97	0.94	1	1
	F-Measure	0.96	0.97	0.97	0.95	0.97	0.99
Training Time (sec)		0.01	0.03	0.07	121.59	4336.37	3763.62

Table 4. T-Test results

	SBB “Last-Best”	SBB “Most-Frequent”
Pruned C4.5	0.11	0.50

5 Conclusion and Future Work

Legitimate users are not the only ones that use DNS to communicate. Modern botnets avoid ‘hardcoding’ the address of the C&C server because if the C&C server is identified, they can be blocked at the firewall level. DNS provides a scalable solution for botnets since a list of domain names can be passed to the victim host as C&C server. As long as the victim manages to connect to the server using one of the domains in the list, it will download the malware and join the botnet. On the other hand, all the domains on the list (whether they resolve to an IP address or not) need to be blacklisted to be able to fully mitigate the attack.

Fortunately for the defenders, DNS traffic of a botnet exhibits abnormal properties that can be detected. The most important property is the structure of the domains that are being queried, i.e. long, with many sub-domains and seemingly random set of characters. Thus, a suitable solution is to monitor the communications at the DNS level to detect abnormal query patterns, specifically queries that a human would not possibly be able to type, based on temporal, structural and syntactic properties.

In this work an evolutionary computation based solution is investigated. To this end, we designed and developed the Stateful-SBB classifier, which is utilized to support variable length input. In addition to providing a very high accuracy on classification and automatically generating signatures, Stateful-SBB identifies the set of attributes to be used in classification automatically without requiring any a priori knowledge, whereas typical classifiers evaluated requires a fixed set of features extracted based on a priori knowledge. The results show that Stateful-SBB based system performs comparable to other classification methods without requiring a feature set to be determined a priori. Future work will include improvement on the training time of the Stateful-SBB, evaluating other classifiers such as SVM and testing the proposed system under other data sets.

Acknowledgments. This research is supported by the Natural Science and Engineering Research Council of Canada (NSERC) grant, and is conducted as part of the Dalhousie NIMS Lab at http://projects.cs.dal.ca/projectx/.

References

1. Feily, M., Shahrestani, A.: A Survey of Botnet and Botnet Detection. Emerging Security Information. In: Emerging Security, Systems and Technologies (2009)
2. Damballa Inc.:Top 10 Botnet Threats (2010), `http://www.damballa.com`
3. McAfee Labs Thread Reports, `http://www.mcafee.com/apps/view-all/publications.aspx`
4. Doucette, J., McIntyre, A.R., Lichodzijewski, P., Heywood, M.I.: Symbiotic Coevolutionary Genetic Programming: A Benchmarking Study Under Large Attribute Spaces. Genetic Programming and Evolvable Machines 13(1), 71–101 (2012)
5. Vuong, S.T., Alam, M.S.: Advanced Methods for Botnet Intrusion Detection Systems. In: Intrusion Detection Systems. InTech. (2011)
6. Bailey, M., Cooke, E., Jahanian, F., Xu, Y., Karir, M.: A Survey of Botnet Technology and Defense. In: CATCH 2009 (2009)
7. The Role of DNS in Botnet Command & Control. In: Open DNS Inc., Whitepaper (2012)
8. Zhang, L., Yu, S., Wu, D., Watters, P.: A Survey on Latest Botnet Attack and Defence. In: TrustCom, pp. 53–60 (2001)
9. Stalmans, E., Irwin, B.: A Framework for DNS Based Detection and Mitigation of Malware Infections on a Network. In: Information Security South Africa (2011)
10. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting Algorithmically Generated Domain-Flux Attacks With DNS Traffic Analysis. IEEE/ACM Transaction on Networking 20, 1663–1977 (2012)

11. Ma, J., Saul, L.K., Savage, S., Voelker, G.: Beyond blacklists: Learning to detect malicious Web sites from suspicious URLs. In: ACM KDD (2009)
12. Antonakakakis, M., Perdisci, R., Dagon, D.: Building a Dynamic Reputation System for DNS. In: USENIX Security 2010 (2010)
13. Alpaydin, E.: Introduction to Machine Learning. MIT Press (2004)
14. Abuse: AMaDA, `https://palevotracker.abuse.ch/`
15. Abuse: Zeus Tracker, `https://zeustracker.abuse.ch/`
16. Alexa, `http://www.alexa.com/topsites`
17. Top Level Domain Names, `http://mxr.mozilla.org/mozilla-central/source/netwerk/dns/effective_tld_names.dat?raw=1`
18. WEKA, `http://www.cs.waikato.ac.nz/ml/weka/`
19. Lichodzikewski, P., Heywood, M.I.: Symbiosis Complexification and Simplicity under GP. In: GECCO 2010 (2010)