Towards Automatic and Lightweight Detection and Classification of Malicious Web Contents

Aziz Mohaisen
Department of Computer Science and Engineering
State University of New York at Buffalo

***Abstract*—Malicious webpages are today one of the most prevalent threats in the Internet security landscape. To understand such problem, there has been several efforts of analysis, classification, and labeling of malicious webpages, ranging from the simple static techniques to the more elaborate dynamic techniques. Building on such efforts, this work summarizes our work in the design and evaluation of a system that utilizes machine learning techniques over network metadata to identify malicious webpages and classify them into broader classes of vulnerabilities. The system uses easy to interpret features, utilizes uniquely acquired dynamic network artifacts, and known labels for rendered webpages in a sandboxed environment. We report on the success (and failure) of our system, and the way forward by suggesting open directions for practical malicious web contents classification.**

I. Introduction

Today, web threats are a great part of the overall security landscape, where malware authors use web technologies to effectively distribute their malice and target their victims. This trend has been reported in the growing number of malicious scripts and exploits on the web. For example, in 2014 alone, Kaspersky lab reported more than 123 Million unique malicious objects, and 1.9 Million of which were targeted against banking services, which are of the utmost sensitivity nature [1]. Such trend calls for better and efficient malware analysis, detection, and classification algorithms. To this end, antivirus vendors and security threat analysis providers invest substantial efforts and resources to develop analysis techniques for addressing this issue. Those techniques range in the technologies they use, and their level of sophistication. Broadly, they are classified into three types: static, dynamic, and hybrid techniques. Despite their computational efficiency that makes an excellent candidate solution, static solutions have various drawbacks, and mostly importantly their low accuracy. On the other hand, dynamic techniques which achieve high accuracy have computational cost, preventing them from wide deployment to address the problem at hand. Hybrid techniques grew to be a better candidate that capitalizes on strengths of both approaches.

The approach reported in this paper follows the hybrid approach to malicious web contents analysis and classification. The hybrid component we follow in this approach makes use of intelligence and reputation information (static) for better scalability: deep analysis for dynamic features extraction is coupled with such static features for better accuracy (than static alone) and better scalability (than dynamic alone). The amount of dynamic features collected and used in the resulting approach is calibrated as a cost parameter.

The broad motivation of the work reported in this paper is as follows. First, existing techniques for highly accurate malicious web contents analysis and classification, including the predecessor of the system reported in this paper, are computationally expensive. Part of the cost associated with such systems is not only that they use dynamic analysis that requires execution of web contents in sandboxed environments, but that they are also not particularly focused on reaching the end goal of the problem at hand; analysis and classification. Those systems often are intended for providing context of attacks by, for example, understanding related artifacts through the analysis of individual system calls and dependencies, or even performing deep packet inspection. Second, while they provide a promising level of accuracy, existing systems in the literature for malicious web contents analysis rely mainly on static features [8], [19], [20], [24], [26]. While efficient, we theorize that their reliance on the static features could be a critical point of attack: those features could perhaps be manipulated to evade detection. To this end, we envision a scenario where the use of a mixed set of features, both dynamic and static, could make it harder for the adversary to evade detection, and improve both the detection and classification accuracy at a reasonable additional cost. To reach such goals, various challenges arise, which we outline in this paper.

The contributions of this work are as follows. First, we present a system that identifies whether web contents are malicious or benign utilizing a mixed set of dynamic and static features. Those features are carefully selected and extracted to achieve scalability properties for the introduced system. Second, we evaluate the system on a real-world dataset that contains multiple variants of vulnerabilities, and show its ability to distinguish between them. Third, by showing shortcomings of the system—which pertain to other systems that share the same underlying design concepts with ours, we highlight open directions worth further investigation.

The rest of this paper is organized as follows §II presents some details on system-α, which generates the data we use in Δ and the ground truth, while §III presents the

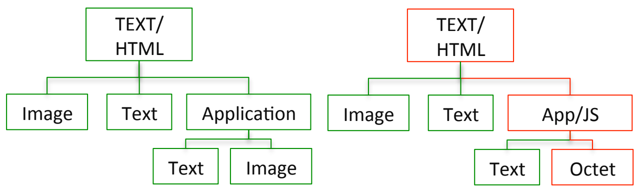

Figure 1. Two examples of transferred file trees (TFTs). The left-side example highlights a benign TFT while the right side example highlights a malicious TFT (through the red branch).

architecture of the Δ system. §IV presents the evaluation of Δ and discusses the results. §V discusses the related work. Finally §VI presents the conclusions and sheds some light on future work.

II. System-α: Dynamic Analysis

The system-α is used for analyzing web contents in a dynamic fashion, and features extracted from such analysis are used for classification of webpage pages into benign or malicious. The system-α crawls websites using an orchestrated and virtualized web browser. For each analyzed webpage, the system maintains records of each HTTP request-response pair made while rendering that page. The system applies static and dynamic analysis techniques to inspect each object retrieved while visiting the webpage, and monitors any changes to the underlying system to decide whether the retrieved object is malicious or not.

The basic object analyzed by system-α is called transferred files (TFs). Determining whether an webpage is malicious or not depends on the retrieved and analyzed TFs. system-α labels TFs into benign or malicious, and if malicious they are labeled based the type of malice associated with them and uncovered by system-α.

The types associated with files by system-α are one or more of the following. (i) Injection type, which occures when a website is compromised, allowing an attacker to add arbitrary HTML and javascript to the legitimate content of the site with the purpose of referencing malicious content. (ii) Exploit type, which implies that an exploit code for a vulnerability in the browser or browser helper was found—often used for drive-by downloads. (iii) Exploit kit type, which is a collection of exploits bundled together and usually sold in black market. (iv) Obfuscation type, which is assigned when a TF contains obfuscated code with known malicious activity behavior. (v) Defacement type, which occurs when an attacker hacks into a website and replaces some content indicating that the site has been hacked into. (vi) Redirection type, which is assigned when a TF redirects to a known malicious content. (vii) Malicious executable or archive type, which means that either an executable or an archive file that contains malicious code was detected to be downloaded by visiting the webpage. (viii) Server side backdoor type, which is assigned when a TF shows symptoms of being a known server-side backdoor script.

The processing of the data of each URL by system-α results in a tree-like structure (Fig. 1) where each node represents a TF. Each node stores basic file attributes and network information (HTTP response code, IP address, and AS number, etc.). These nodes also contain classification data from system-α's deep analysis and we use this as ground-truth in evaluating our proposed approach, Δ.

III. Δ: Hybrid System for Web Contents

There are multiple challenges that Δ tries to address summarized as follows. (i) Webpages face different types of vulnerabilities: exploit kits, defacement, malicious redirections, code injections, and server-side backdoors – all with different signatures, as mentioned in section II. (ii) Malice in web contents may not even be the fault of a webpage owner, as it is the case with advertisement networks that exhibit such behavior. (iii) The distribution of behavior is highly imbalanced, with typical dataset having 40 times more benign objects than malicious ones.

Despite these challenges, we show that Δ is currently capable of achieving 96% accuracy, with injection attacks and server-side backdoors being identified as areas for performance improvement and future attention. The system is also capable of identifying the types of detected vulnerabilities with exact match in 91% of the cases, with a difference of 1 and 2 labels in 6% and 3% of the cases.

Design goals. To this end, there are two basic ideal goals for the proposed system Δ. (i) The main goal of Δ is to identify whether a webpage is malicious or not based on the basic metadata maintained by system-α, without having to compute any complex and expensive features. (ii) If a webpage is classified as malicious, the system also aims at identifying which type of malice it has.

Design. The layout of the system is as follows (with more details in [18]). The system receives artifacts of data generated by the system described in section II. The system works in training and operational modes, described as follows. The system is trained by labeled webpages, in which each individual TF is labeled as benign, or malicious. The system uses the basic meta-data stored in the system, in addition to a set of simple features generated based on those attributes. This generation is handled by the feature generation module which uses IP and WHOIS databases to acquire information about the IP address and the domain name of the associated TF. After the feature generation stage, the data is preprocessed, and features are filtered using a feature selection module, before the data is sent to the classification modules. Then, a two-stage classification procedure is trained based on the preprocessed data.

In the operational mode, for an unlabeled webpage, the system transforms each TF into a feature vector as done by the feature generation module in the training phase, and then the features are preprocessed and filtered based on the feature selection results from the training phase. The TF is then labeled with the label most close to it in the vector space based on a highly accurate ground truth (human vetted in many cases). The following subsections provide further details on features selection and classification.

A. Classification Features

To achieve the design goals in section III, Δ relies on a rich set of mixed static and dynamic features, and uses nearly 40 basic features for the classification process. The features fall in the following broad categories.

META-DATA FEATURES: feature set that includes HTTP header information, such as HTTP method, response code, compression mode, etc. This also includes the AS number, and retrieved file (outcome of libmagic).

URI FEATURES: features derived from the URI associated with a TF including basic lexical statistics; lengths (hostname, path and query), dot count, slash count, special characters ratio and the average path segment length. This also includes binary features of whether the URI contains an explicit IP, or an explicit port number.

TF FEATURES: features that we extract from the TF-tree to capture the relationship between different TFs that belong to a single webpage. The TF-tree features capture parent-child host/IP diversity, TF depth, number of children and the child-parent type relationship.

DOMAIN NAME FEATURES: features derived from the domain name of the URI of the TF, including the registrar's id and age information, e.g. creation data and expiration date.

IP-BASED FEATURES: features derived from the IP address associated with the TF, including geo location; country, city and region, and domain or organization for which the IP is registered. Two IP prefixes (/24 and /28) are considered as features to identify networks with malicious activity.

B. Features Preprocessing

After the feature values for each category are inferred, a preprocessing stage is needed before forwarding this data to the classifiers for training and testing purposes. The preprocessing is done based on the feature type. For example, for numeric features, such as the lexical counts, proper scaling (normalization) is applied to the feature to keep the values between 0 and 1. For categorical features such as the top-level domain name or AS number, among others, we apply feature binarization, in which a binary feature is introduced per each possible value, since the feature cannot be encoded numerically due to the lack of order between values. This approach has been employed in literature, as in [19] and has shown promise. However, we note that such technique would certainly result in high-dimensional feature vectors that require a scalable classifier suitable for high dimensionality vectors, albeit a feature selection process would reduce dimensionality for alternative techniques.

FEATURE SELECTION: Due to the high dimensional feature vectors, reducing dimensionality through a feature selection technique would facilitate operational deployment of our technique. Therefore, in our experiments, we study the effect of reducing the dimensionality through a χ^2 metric.

C. Classification

Two-stage classification process is employed. The first classification stage includes a binary classifier that is trained with all the TFs from benign and malicious samples. We use an SVM classification algorithm based on Stochastic Gradient Descent using L1-norm for this stage. In the second stage, we build another binary classifier for each type of vulnerability. Each classifier in the second stage is trained using the malicious TF data only, e.g. the injection classifier is trained by the data containing (injection TFs versus No injection but malicious TFs).

We employ this two-stage model because of the limitations of other possible approaches. For example, a multi-class classifier will not capture the observation that some TFs are labeled with more than one label. Additionally, we found that using multiple binary classifiers directly in a single stage, where each classifier is trained for only one type of attack—versus all the other benign and remaining malicious TFs—will lead to lower accuracy and a higher training time. The low accuracy in this case is due to the higher possibility of false positives because of using multiple classifiers at once. Therefore, we propose this two-stage model to filter out the malicious TFs first using a global classifier, then identify the type of malice separately.

In operation phase, whenever a webpage is analyzed, the data of each TF retrieved while visiting the URL are used to predict whether it is malicious or not. A URL is labeled as benign if all retrieved TFs were benign by the classification algorithm. Then, the type of malice is identified through the second stage if the TF was labeled as malicious.

IV. EVALUATION

We present the evaluation and analysis of Δ. We give an overview and description of the dataset with the evaluation procedure and metrics. Then, we introduce the performance of the binary classification and label prediction, followed by the effect of feature selection on the system accuracy.

DATASET. The dataset we used consists of 20k webpages, 10k each of "malicious" and "benign" types. These URLs were randomly selected from system-α operational history of Internet-scale crawling. As mentioned earlier, system-α labels the webpages using sophisticated static and dynamic analysis techniques, and hence we consider such labels as our ground truth labels. Analyzing the URLs of the dataset yields 800k benign TFs and 20k malicious TFs. Each webpage contains about 40 TFs on average. For the malicious

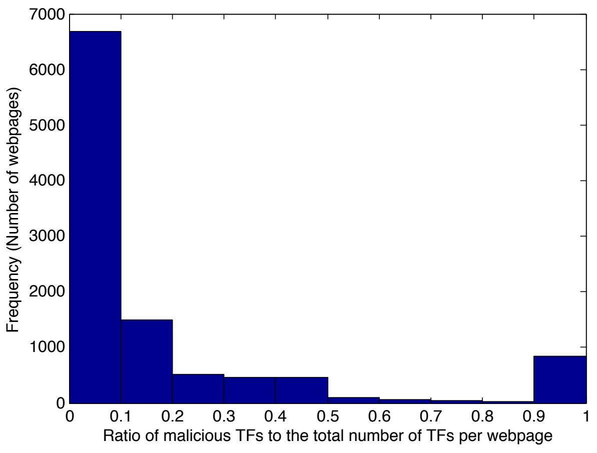

Figure 2. Malicious TFs per malicious webpages

Table I
DISTRIBUTION OF MALICE AMONG THE TFS

Type	# samples	Type	# samples
Injection	10696	Injection/exploit	181
Exploit kit	5102	Archive	31
Redirection	1657	Backdoor	25
Exploit	1030	Injection/redirection	15
Executable	800	Injection/defacement	5
Defacement	432	Injection/obfuscation	2

webpages, a histogram of the percentage of the number of malicious TFs per each malicious webpage is shown in Fig. 2. The figure shows that for most malicious webpages, less than 10% of the retrieved TFs are malicious. This confirms the intuition we have for building the classifiers based on individual TFs.

The system-α labels each TF according to type of malice it uncovered. Note that a malicious TF may be labeled with more than one label at the same time. That is a reason a classifier was built for each malice type in the label prediction module. The distribution of vulnerabilities among the malicious TFs can be illustrated in detail through Table I.

A. *Evaluation Procedure and Metrics*

The evaluation of the system was conducted using 10-fold cross-validation. For consistency, the dataset was partitioned randomly in a way that guarantees that the distribution of number of TFs per webpage is roughly maintained, so that the total number of TFs per partition is almost the same. The performance metrics are considered at TF and webpage granularity, with more focus on the latter since this is the end system goal. Recall that a webpage is labeled as malicious if any of its TFs was labeled by the classifier as malicious. The metrics considered for the evaluation are mainly the false positives rate, the false negatives rate, and the F1-score, defined in terms of the precision and recall. Precision and recall are as in the literature.

B. *Binary Classification Performance*

We start by describing the results of the first classification stage, which aims to identify whether a webpage is malicious or benign, only. Table II enumerates the performance metrics at both TF and webpage granularity, showing an overall result of 7.6% FN rate and 6.3% FP rate for the webpage results. The reason for having a 14.7% FN rate on the TF-level is that simple metadata may not be indicative for all types of TF malice behavior. Additionally, compared to previous literature, the TF results are consistent with respect to the fact that our TF records dataset is highly imbalanced. Literature studies showed that as the data gets highly imbalanced, the accuracy degrades, e.g. 25% FN rate at a ratio of 100:1 of benign to malicious URLs [20].

Table II
BINARY CLASSIFICATION RESULTS

	Prec.	Recall	F-score	FP	FN
TF-level	0.390	0.852	0.530	0.0314	0.147
page-level	**0.935**	**0.924**	**0.930**	**0.063**	**0.076**

To understand how well the detection performs, Fig. 4 shows the detection rate per each vulnerability/attack type at the TF-level, which describes the ratio of the TFs labeled as malicious successfully. Note that the “injection” and “server side backdoor cases” were most detrimental to overall performance. This is made clear in Tab. III which provides overall performance without those problematic instances, resulting in 2.5% FP rate and 4.8% FN rate overall.

Table III
BINARY CLASSIFICATION W/O “INJECTION”

	Prec.	Recall	F-score	FP	FN
TF-level	0.527	0.873	0.657	0.0153	0.126
page-level	**0.948**	**0.951**	**0.949**	**0.0257**	**0.048**

C. *Label Prediction Performance*

After a TF is labeled as malicious by the system, the system labels it according to the type of attack/malice it carries by the label prediction module described earlier in Sec. III. In this section, the results of this module are presented. The main metric we used for the evaluation of the label prediction is the number of different labels between the ground truth and the predicted ones. As an example for illustration, if the ground truth is {Injection}, and the system labeled the malicious TF as {Injection, Exploit}, then this is considered a difference of one. If the predicted label was only {Exploit}, this is considered a difference of two, since two changes are necessary to make the prediction correct. Fig.3 illustrates the CDF of the label difference metric. As the figure clearly shows, the median of the difference in label predictions is zero. In fact in more than 90% of the cases, there was no difference between the predicted labels and the ground truth, and in only about 3% of the cases there was a difference of two labels.

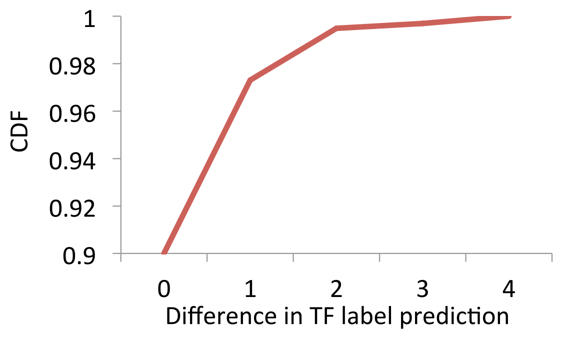

Figure 3. The CDF of the difference in malice label predictions

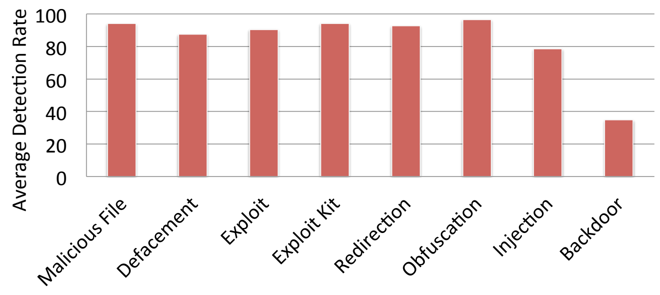

Figure 4. Detection rate per TF vulnerability type for various malice types

We evaluate the capability of each label prediction classifier (figures omitted). In this experiment, the server side backdoor classifier had the highest miss-label rate, perhaps due to the few samples of server side backdoors in the dataset (Table I). Also, both the exploit and exploit kit classifiers have high miss-label rates as well, which suggests that new exploit attacks that the system did not specifically learn about may not be directly easy for the system to infer. With respect to the wrong-label rate, one interesting observation is that the injection classifier had the highest wrong-label rate. This could be because most of the malicious TFs in the dataset are Injection attacks (Table I), which could have resulted tendency towards labeling malicious TFs as injection due to the imbalanced training dataset.

D. *Feature Selection*

Due to the number of categorical features we have, we have high dimensional vectors result due to binarization, which affects scalability. To address this issue, we employ feature selection. Using the χ^2 score we calculated for each feature, we observed that the feature scores considerably vary, ranging from 10^{-5} to 10^{5}. This may suggest that half of the features may not be very important for the classification. We confirmed that by studying the effect of the number of features on detection in Δ. In this confirmation (omitted), we show that the performance increases rapidly till it reaches a turning point of a stable F-score.

V. Related Work

Various efforts have been on the problem at hand, although differing from our work in various aspects, including features richness, quality of labels, and their context. Most closely related to our work are the works in [8], [19], [20], [24], [26], although differing in using static analysis-related features in reaching conclusions on a webpage. On the other hand, Δ relies on utilizing simple features extracted from the dynamic execution of a webpage and loading its contents in a sandboxed environment, with the goal of incorporating that as a tiered classifier in system-α.

Table IV
Distribution of Generated Features

Feature Category	Number of Features
Meta-data based	18850
URL-based	378740
TF tree-based	157
Domain name-based	419
IP-based	65153

Some related work using structural properties of URLs in order to predict malice include the works in [11], [19], [27] for email spam, and in [7], [21] for phishing detection. Additionally, using domain registration information and behavior for malware domain classification was explored in [15], [19], [25]. Related to that is the work on using machine learning techniques to infer domains behavior based on DNS traces. Bilge et al. proposed Exposure [6], a system to detect malware domains based on DNS query patterns on a local recursive server. Antonakakis et al. [3] functions similarly but analyzes global DNS resolution patterns and subsequently creates a reputation system for DNS atop this logic [2]. Gu et al. [12]–[14] studied several botnet detection systems utilizing the same tools of DNS monitoring. Dynamic malware analysis and sandboxed execution of malware were heavily studied in the literature, including surveys in [9], [10]. Bailey et al. [4] and Bayer et al. [5] have focused on behavior-based event counts. Feature development has since advanced such that malware families can now be reliably identified [17] and dynamic analysis can be deployed on end hosts [16]. Finally, network signature generation for malicious webpages is explored in [22], [23] for drive-by-download detection.

VI. Conclusion And Open Directions

This paper presented Δ, a system that uses machine learning over simple network artifacts that are inferred during dynamic webpage execution. Δ's goal is to detect whether a webpage is malicious or not, and to identify the type of malice if the webpage was found malicious. Under cross-validation and a dataset that spans different types of attack behavior, the system was able to detect malicious webpages with an accuracy of 93% identifying injection and derver-side backdoor vulnerabilities as the main areas requiring detection improvement. Excluding injection samples from the dataset has resulted in an accuracy reaching 96%. Additionally, the malice labeling module was able to detect the label(s) of malicious TFs exactly in about 91% of the cases, with a difference of one and two labels in 6% and 3% of the cases respectively.

Several directions can be explored. (i) Since many of the features have a dynamic nature over time, e.g., IP, URI's, etc., one direction is to explore adaptive mechanisms to capture such dynamic changes over time. (ii) Further studies need to be focused on enhancing the accuracy models presenting in this paper to detect injection and server side backdoor attacks, in addition to identify exploit attacks, by incorporating context information. (iii) Scoring of malice is worth investigating, not as a binary class but as a degree of severity, and ways to incorporate that in the classifier. (iv) While generated by system-α, they are not used; payload and content-based features derived from Javascript as in [8], [26], or flow information features as in [26] can be extracted and utilized for improving the accuracy of classification.

ACKNOWLEDGEMENT. The author thanks Ahmed Kosba for his contribution to basis of this work in [18].

REFERENCES

[1] —. Kaspersky security bulletin 2014. overall statistics for 2014. http://bit.ly/1D46aS2, 12 2014.

[2] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *USENIX Security*, 2010.

[3] M. Antonakakis, R. Perdisci, W. Lee, N. V. II, and D. Dagon. Detecting malware domains at the upper DNS hierarchy. In *USENIX Sec. Sym.*, 2011.

[4] M. Bailey, J. O. J. Andersen, Z. Mao, F. Jahanian, and J. Nazario. Automated classification and analysis of Internet malware. In *RAID*, 2007.

[5] U. Bayer, P. M. Comparetti, C. Hlauschek, C. Krügel, and E. Kirda. Scalable, behavior-based malware clustering. In *NDSS*, 2009.

[6] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *NDSS*, 2011.

[7] A. Blum, B. Wardman, T. Solorio, and G. Warner. Lexical feature based phishing URL detection using online learning. In *AISec*, 2010.

[8] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: a fast filter for the large-scale detection of malicious web pages. In *WWW*, 2011.

[9] J. Chang, K. K. Venkatasubramanian, A. G. West, and I. Lee. Analyzing and defending against web-based malware. *ACM Computing Surveys*, 45(4), 2013.

[10] M. Egele, T. Scholte, E. Kirda, and C. Kruegel. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys*, 44(2), 2008.

[11] M. Felegyhazi, C. Kreibich, and V. Paxson. On the potential of proactive domain blacklisting. In *LEET*, 2010.

[12] G. Gu, R. Perdisci, J. Zhang, and W. Lee. BotMiner: Clustering analysis of network traffic for protocol and structure independent botnet detection. In *USENIX Security*, 2008.

[13] G. Gu, P. Porris, V. Yegneswaran, M. Fong, and W. Lee. Bothunter: Detecting malware infection through IDS-driven dialog correlation. In *USENIX Security*, 2007.

[14] G. Gu, J. Zhang, and W. Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *NDSS*, 2008.

[15] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck. Understanding the domain registration behavior of spammers. In *IMC*, 2013.

[16] C. Kolbitsch, P. M. Comparetti, C. Kruegel, E. Kirda, X. Zhou, and X. Wang. Effective and efficient malware detection at the end host. In *USENIX Sec Symposium*, 2009.

[17] D. Kong and G. Yan. Discriminant malware distance learning on structural information for automated malware classification. In *KDD*, 2013.

[18] A. E. Kosba, A. Mohaisen, A. G. West, T. Tonn, and H. K. Kim. ADAM: automated detection and attribution of malicious webpages. In *WISA*, pages 3–16, 2014.

[19] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *KDD*, 2009.

[20] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Learning to detect malicious urls. *ACM Trans. Intell. Syst. Technol.*, (3):30:1–30:24, May 2011.

[21] D. K. McGrath and M. Gupta. Behind phishing: An examination of phisher modi operandi. In *LEET*, 2008.

[22] N. Provos, P. Mavrommatis, M. A. Rajab, and F. Monrose. All your iFRAMEs point to us. In *USENIX Security*, 2008.

[23] N. Provos, D. McNamee, P. Mavrommatis, K. Wang, N. Modadugu, et al. The ghost in the browser analysis of web-based malware. In *HotBots*, 2007.

[24] K. Thomas, C. Grier, J. Ma, V. Paxson, and D. Song. Design and evaluation of a real-time url spam filtering service. In *IEEE Security and Privacy*, 2011.

[25] A. G. West and A. Mohaisen. Metadata-driven threat classification of network endpoints appearing in malware. In *Detection of Intrusions and Malware, and Vulnerability Assessment - 11th International Conference, DIMVA 2014, Egham, UK, July 10-11, 2014. Proceedings*, pages 152–171, 2014.

[26] L. Xu, Z. Zhan, S. Xu, and K. Ye. Cross-layer detection of malicious websites. In *CODASPY*, 2013.

[27] T.-F. Yen, A. Oprea, K. Onarlioglu, T. Leetham, W. Robertson, A. Juels, and E. Kirda. Beehive: Large-scale log analysis for detecting suspicious activity in enterprise networks. In *ACSAC*, 2013.