Efficient and Accurate Behavior-Based Tracking of Malware-Control Domains in Large ISP Networks

BABAK RAHBARINIA, Auburn University Montgomery
ROBERTO PERDISCI, University of Georgia, Georgia Institute of Technology
MANOS ANTONAKAKIS, Georgia Institute of Technology

In this article, we propose *Segugio*, a novel defense system that allows for efficiently tracking the occurrence of new *malware-control* domain names in very large ISP networks. Segugio passively monitors the DNS traffic to build a machine-domain bipartite graph representing *who is querying what*. After labeling nodes in this *query behavior* graph that are known to be either benign or malware-related, we propose a novel approach to accurately detect previously unknown malware-control domains.

We implemented a proof-of-concept version of Segugio and deployed it in large ISP networks that serve millions of users. Our experimental results show that Segugio can track the occurrence of new malware-control domains with up to 94% true positives (TPs) at less than 0.1% false positives (FPs). In addition, we provide the following results: (1) we show that Segugio can also detect control domains related to new, previously unseen malware families, with 85% TPs at 0.1% FPs; (2) Segugio's detection models learned on traffic from a given ISP network can be deployed into a different ISP network and still achieve very high detection accuracy; (3) new malware-control domains can be detected days or even weeks before they appear in a large commercial domain-name blacklist; (4) Segugio can be used to detect previously unknown malware-infected machines in ISP networks; and (5) we show that Segugio clearly outperforms domain-reputation systems based on Belief Propagation.

CCS Concepts: • **Information systems → Information systems applications**; **Data mining**; • **Security and privacy → Malware and its mitigation**; **Network security**; • **Computing methodologies → Machine learning approaches**; **Learning in probabilistic graphical models**

Additional Key Words and Phrases: Behavioral analysis, graph mining, malware-control domains

ACM Reference Format:
Babak Rahbarinia, Roberto Perdisci, and Manos Antonakakis. 2016. Efficient and accurate behavior-based tracking of malware-control domains in large ISP networks. ACM Trans. Priv. Secur. 19, 2, Article 4 (August 2016), 31 pages.
DOI: http://dx.doi.org/10.1145/2960409

This material is based in part on work supported by the National Science Foundation, under grant no. CNS-1149051, and the Department of Homeland Security, under contract no. N66001-12-C-0133. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Department of Homeland Security.
This work is also partially supported by a grant from the Auburn University at Montgomery Research Grant-in Aid Program.
Authors' addresses: B. Rahbarinia, Auburn University at Montgomery, Math and Computer Science Dept., 7061 Senators Dr., Montgomery, AL 36117; email: brahbari@aum.edu; R. Perdisci, University of Georgia, Computer Science Dept., 415 Boyd GSRC, Athens, GA 30602; email: perdisci@cs.uga.edu; M. Antonakakis, Georgia Institute of Technology, School of Electrical and Computer Engineering, Klaus Advanced Computing Building, 266 Ferst Drive, Atlanta, GA 30332; email: manos@gatech.edu.

1. INTRODUCTION

Despite extensive research efforts, malicious software (or *malware*) is still at large. In fact, numbers clearly show that malware infections continue to be on the rise [Symantec 2013a, 2013b]. Because malware is at the root of most of today's cybercrime, it is of utmost importance to persist in our battle to defeat it or, at the very least, to severely cripple its ability to cause harm by tracking and blocking its command-and-control (C&C) communications.

Our Research. In this article, we propose *Segugio*[1], a novel defense system that allows for efficiently tracking the occurrence of new *malware-control* domain names in very large ISP networks. We define a malware-control domain as a domain that provides a channel for attackers to control malware's critical functionalities. Accordingly, we consider as malware-control domains those domains that are used exclusively to contact machines that, in turn, provide commands or updates to the malware. In essence, malware-control domains provide "malware-only" functionalities. Segugio automatically learns how to discover new malware-control domain names by monitoring the DNS *query behavior* of both known malware-infected machines as well as benign (i.e., "non-infected") machines. Our work is based on the following simple, but fundamental, intuitions: (1) in time, as the infections evolve, infected machines tend to query new malware-control domains; (2) machines infected with the same malware or, more precisely, malware family, tend to query the same (or a partially overlapping) set of malware-control domains; and (3) benign machines have no reason to query malware-control domains that exist for the sole purpose of providing malware C&C capabilities or other "malware-only" functionalities.

Segugio's main goal is to *track current malware infections* to discover where (i.e., to what new names) malware-control domains relocate. In addition, we will show that Segugio can also discover malware-control domains related to *new malware families* previously unseen in the monitored networks.

To put these observations and goals into practice, we propose an efficient strategy. First, Segugio passively observes the DNS traffic between the users' machines and the ISP's local DNS resolver to build an annotated bipartite graph representing *who is querying what*, as shown in Figure 1. In this graph, nodes represent either machines or domain names, and an edge connects a machine to a domain if that machine queried the domain during the considered traffic observation time window. The domain nodes are augmented with a number of annotations, such as the set of IPs that a domain resolved to, its domain activity (e.g., how long ago a domain was first queried), and so on. Then, we label as *malware* those nodes that are already known to be related to malware control functionalities. For example, we can first label known malware C&C domains, and as a consequence also propagate that label to the machines by marking any machine that queries a C&C domain as malware infected. Similarly, we can label as *benign* those domains that belong to a whitelist of popular domains (e.g., according to alexa.com) and, consequently, propagate the *benign* label to machines that query *exclusively* known benign domains. All remaining machine nodes are labeled as *unknown*, because they do not query any known malware domain and query at least one *unknown* domain, whose true nature is not yet known. Segugio aims to efficiently classify these *unknown* graph nodes.

Approach. Based on the machine-domain bipartite graph (Figure 1), we can observe that *unknown* domains that are consistently queried only (or mostly) by known malware-infected machines are likely themselves malware related, especially if they have been active only for a very short time or point to previously abused IP space. In essence, we combine the machines' query behavior (i.e., who is querying what) with a

[1]The name *Segugio* refers to an Italian hound dog breed.

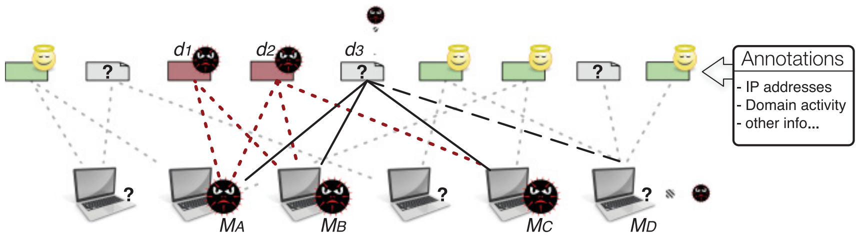


Fig. 1. Machine-domain annotated graph. By observing *who is querying what*, we can infer that d_3 is likely a malware-related domain and, consequently, that M_D is likely infected.

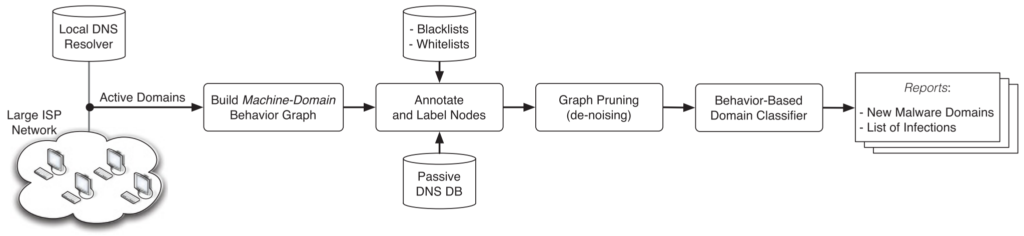


Fig. 2. Segugio system overview.

number of other domain-name features (annotated in the graph) to compute the probability that a domain name is used for malware control or that a machine is infected.

Main Differences with Regard to Previous Work. Recently, researchers have proposed domain-name reputation systems [Antonakakis et al. 2010; Bilge et al. 2011] as a way to detect malicious domains by modeling historic domain-IP mappings, using features of the domain-name strings, and leveraging past evidence of malicious content hosted at those domains. These systems mainly aim to detect malicious domains in general, including phishing, spam domains, and the like.

Note that, while both Notos [Antonakakis et al. 2010] and Exposure [Bilge et al. 2011] leverage information derived from domain-to-IP mappings, they do not leverage the query behavior of the machines “below” a local DNS server. Unlike Antonakakis et al. [2010] and Bilge et al. [2011], our work focuses specifically on accurately tracking new “malware-only” domains by monitoring the DNS *query behavior* of ISP network users. In Rahbarinia et al. [2015], we show that our approach yields both a lower false-positive rate and much higher true positives compared to Notos [Antonakakis et al. 2010] (in Rahbarinia et al. [2015], we perform a direct comparison to a version of Notos provided by the original authors of that system).

Kopis [Antonakakis et al. 2011] has a goal more similar to ours: detect malware-related domains. However, Kopis's features (e.g., *requester diversity* and *requester profile*) are engineered specifically for modeling traffic collected at authoritative name servers, or at top-level-domain (TLD) servers, thus requiring access to authority-level DNS traffic [Antonakakis et al. 2011]. This type of global access to DNS traffic is extremely difficult to obtain, and can only be achieved in close collaboration with large DNS zone operators. Furthermore, due to the target deployment location, Kopis may allow for detecting only malware domains that end with a specific TLD (e.g., .ca). Unlike Kopis, Segugio allows for efficiently detecting new malware-control domains, regardless of their TLD, by monitoring *local* ISP traffic (e.g., DNS traffic between ISP users and their local DNS resolver). Therefore, Segugio can be independently deployed by ISP network administrators without the need of a collaboration with external DNS operators.

Another work related to ours is Manadhata et al. [2014], which uses graphical models to detect malicious domains via loopy belief propagation [Koller and Friedman

2009]. Unlike Manadhata et al. [2014], which is limited to using machine–domain relationships, Segugio can complement information about the query behavior of the machines with properties of the queried domain names (e.g., their lifetime and resolved IP information). This allows us to achieve a significantly higher accuracy compared to Manadhata et al. [2014], especially at low false-positive rates. In addition, the approach in Manadhata et al. [2014] does not scale well to the very large ISP-level DNS traffic that is the target of our work. On the other hand, Segugio is specifically designed to be highly efficient and has been evaluated in multiple very large ISP networks. To concretely compare our own Segugio system to the approach proposed in Manadhata et al. [2014], we implemented a loopy belief propagation algorithm using the GraphLab [Low et al. 2010] distributed computing framework, and performed a number of experiments over the same datasets that we use for evaluating Segugio (see Sections 3 and 6). Our results indicate that Segugio, on average, can achieve 45% better accuracy compared to Manadhata et al. [2014]. In addition, the efficient classification approach that we propose in this article allows us to process an entire day of DNS traffic in minutes, rather than the tens of hours required by loopy belief propagation.

We further discuss the differences between Segugio and other related work in Section 8.

Summary of Our Contributions. In summary, with Segugio, we make the following contributions:

—We propose a novel behavior-based system that can efficiently detect the occurrence of new malware-control domains by tracking the DNS query behavior of malware infections in large ISP networks.

—We implemented a proof-of-concept version of Segugio, and deployed it in three large ISP networks that serve millions of users. Our experimental results show that Segugio, on average, can classify an entire day's worth of ISP-level DNS traffic in just a few minutes, achieving a true positive (TP) rate above 94% at less than 0.1% false positives (FPs).

—We provide the following additional results: (1) we show that Segugio can also detect malware-control domains related to previously unseen malware families, with 85% TPs at 0.1% FPs; (2) Segugio's detection models learned on traffic from an ISP network can be deployed in another ISP network and still achieve very high detection accuracy; (3) new malware-control domains can be detected days or even weeks before they appear in a large commercial domain-name blacklist; (4) new and previously unknown malware-infected machines can be identified in the ISP networks; and (5) we show that Segugio clearly outperforms Notos [Antonakakis et al. 2010] (refer to Rahbarinia et al. [2015] for a detailed comparison to Notos) as well as systems based on a loopy belief propagation algorithm.

This article is an extension of our previous conference paper [Rahbarinia et al. 2015], and includes numerous new experiments, provides new insights, and extends the explanation of experiments to incorporate many more details. Specifically, we extend our previous work with the following additions:

—We devise a novel graph-based cross-validation algorithm, and perform extensive cross-validation experiments to further evaluate the accuracy of Segugio (Section 5.1).

—We perform a completely new set of experiments that demonstrate the impact that Segugio has on the detection of previously unknown malware-infected machines in large ISP networks (Section 5.9).

—We compare Segugio to an alternative detection algorithm based on belief propagation. Specifically, we implement a distributed belief propagation system inspired by

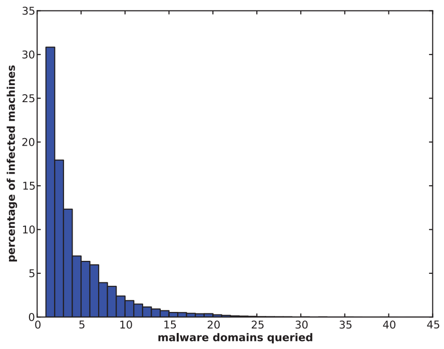


Fig. 3. Distribution of the number of malware-control domains queried by infected machines. About 70% of known malware-infected machines query more than one malware domain.

previous work, and contrast its detection results to Segugio's for a variety of different configurations (Section 6).

—We deployed Segugio in a new ISP network, and added the related traffic to our evaluation data. This addition allowed us to further analyze the performance of our system (Section 5).

—We perform a new set of experiments to further investigate the importance of each group of features used by Segugio (Section 5.4).

—We discuss in detail how cross-day and cross-network experiments are set up and provide further statistical analysis (Sections 5.2 and 5.3).

2. SEGUGIO SYSTEM DESCRIPTION

Segugio's main goal is to track the DNS query behavior of current malware-infected machines to discover their new malware-control domains. In addition, in Section 5.5, we show that Segugio is also capable of discovering domains related to malware families previously unseen in the monitored networks. In this section, we first motivate the intuitions on which our system is based, then describe Segugio's components.

Intuitions. As mentioned in Section 1, Segugio is based on the following main intuitions: (1) in time, infected machines tend to query new malware-related domains; (2) machines infected with the same malware family tend to query partially overlapping sets of malware-control domains; and (3) benign machines have no reason to query domains that exist for the sole purpose of providing "malware-only" functionalities.

We motivate these three intuitions as follows (in reverse order), deferring a discussion of possible limitations and corner cases to Section 7. Intuition (3) is motivated by the fact that most malware-control domains host no benign content whatsoever, because they are often registered exclusively for supporting malware operations. This is particularly true for "recently activated" domains. Therefore, noninfected machines would have no reason to reach out to such domains. Intuition (2) relates to the fact that different variants of a same original malware are semantically similar, thus will exhibit similar network behavior. Finally, intuition (1) is motivated by the fact that malware needs to employ some level of network agility to avoid being trivially blacklisted. To this end, malware-control servers will periodically relocate to new domain names and/or IP addresses. This intuition is further supported by the measurements on real-world ISP-level DNS traffic, reported in Figure 3. During one day of traffic observation, roughly 70% of the malware-infected machines queried more than one malware-control domain name (Figure 3 also shows that it is extremely unlikely that an infected machine queries more than 20 malware-control domains in one day). We also verified that these results are consistent across different observation days and different large ISP networks.

2.1. System Components

We now describe the components of our Segugio system, which are shown in Figure 2.

2.1.1. Machine-Domain Behavior Graph. As a first step, Segugio monitors the DNS traffic between the machines in a large ISP network and their local DNS server, for a given observation time window T (e.g., one day). Accordingly, it constructs a *machine-domain* graph that describes *who is querying what*. Note that we are interested in only authoritative DNS responses that map a domain to a set of valid IP addresses.

Based on the monitored traffic, Segugio builds an undirected bipartite graph $\mathcal{G} = (M, D, E)$ that captures the DNS *query behavior* of machines in the ISP network. Nodes in the set M represent machines, whereas nodes in D represent domain names. A machine $m_i \in M$ is connected to a domain $d_j \in D$ by an edge $e_{ij} \in E$, if m_i queried d_j during the observation window T.

Node Annotations and Labeling. We augment each domain node $d_j \in D$ by recording the set of IP addresses that the domain pointed to during the observation window T (as collected from the live DNS traffic). In addition, we estimate how long ago (with regard to T) the domain was first queried.

We then label machine and domain nodes as either *malware*, *benign*, or *unknown*. Specifically, by leveraging a small number of public and private malware C&C domain blacklists, we can first label known malware-control domains as *malware*. To label benign domains, we leverage the top one million most popular second-level domains according to alexa.com. Specifically, we label as *benign* those domains whose effective second-level domain[2] consistently appeared in the top one million alexa.com list for about 1 year (see Section 3 for details). These domains are unlikely to be used for malware control. Note also that we take great care to exclude certain "free registration" second-level domains from our whitelist, such as dynamic DNS domains and blog domains, because subdomains of these second-level domains can be freely registered and are very often abused. At the same time, we acknowledge that perfectly filtering the whitelist is difficult, and that some amount of noise (i.e., a few malicious domains) may still be present. In Section 5.6, we discuss the potential impact of whitelist noise, which may cause us to somewhat overestimate our false positives.

All remaining domains are labeled as *unknown*, since we do not have enough information about their true nature. These *unknown* domains are the ones that we ultimately want to classify to discover previously unknown malware-control domains. Finally, we label machines as *malware* if they query malware-control domains, in that they are highly likely infected. We can also label as *benign* those machines that query only known benign domains. All other machines are labeled as *unknown*.

2.1.2. Graph Pruning. Because we aim to monitor all DNS traffic in large ISP networks, our machine-domain graph $\mathcal{G}$ may contain several million machine nodes, hundreds of millions of distinct domain nodes, and potentially billions of edges. To boost performance and reduce noise, we prune the graph using the following conservative rules:

(R1) We identify and discard machines that are essentially "inactive" because it is unlikely that they can help our detection system. To be conservative, we filter out only those machines that query $\leqslant 5$ domains.

(R2) In our ISP test networks, we observed a number of machine nodes that likely represent large proxies or DNS forwarders serving an entire enterprise network. Such devices appear as nodes with a very high degree, and tend to introduce substantial levels of "noise." We therefore filter them by discarding all machines

[2]We compute the effective second-level domain by leveraging the Mozilla Public Suffix List (*publicsuffix.org*) augmented with a large custom list of DNS zones owned by dynamic DNS providers.

that query $\geqslant\theta_d$ domains. Empirically setting θ_d to be the 99.99th percentile of the distribution of number of domains queried by a machine was sufficient to remove these outlier machines.

(R3) The graph $\mathcal{G}$ may contain a number of domain nodes that are queried by only one or very few machines. Because we are primarily interested in detecting malware domains that affect a meaningful number of victim machines, we discard all domain names that are queried by *only one* machine.

(R4) Very popular domains, that is, domains that are queried by a very large fraction of all machines in the monitored network, are unlikely to be malware-control domains. For example, assume that we monitor an ISP network serving three million users, in which a domain d is queried by one million of them. If d was a malware-control domain, this would mean that 1/3 of the ISP population is infected with the same malware (or malware family). By extrapolation, this would probably also mean that hundreds of millions of machines around the Internet may be infected with the same malware. While this scenario cannot be completely ruled out, such successful malwares are quite rare. In addition, due to the high number of victims, the malware would draw immediate attention from the security community, likely initiating extensive remediation and takedown efforts. Therefore, we discard all domain names whose effective second-level domain is queried by $\geqslant\theta_m$ machines, where θ_m is conservatively set to 1/3 of all machines in the network, in our experiments.

To make our pruning even more conservative, we apply two small exceptions to these rules. Machines that are labeled as *malware* are not pruned away by rule (R1) even if they query very few domains. The reason is that a machine may appear to be basically "inactive," but the malware running on the machine may periodically query a very small list (e.g., two or three) malware-control domains. We therefore keep those machine nodes, as they may (slightly) help to detect currently unknown malware domains. Note that hosts that are not yet known to be infected and query very few domains ($\leqslant$5 in our experiments) still will be filtered out. However, we believe that special devices that are designed to query a very small and likely very stable set of domains could be more easily monitored via a rule-based or strict anomaly-based detection system, rather than a generic large-scale malware-control detection system such as Segugio. Similarly, known malware-control domains are kept in the graph even if they are queried by only one machine (exception to R3). We also verified, via smaller-scale pilot experiments, that rules R1 through R4 help us to improve both detection accuracy and performance.

2.1.3. Behavior-Based Classifier. We now describe how we measure the features that describe *unknown* (i.e., to-be-classified) domains, which aim to capture the intuitions that we outlined at the beginning of Section 2. Then, we explain how the behavior-based classifier is trained and deployed. We divide the domain features into three groups:

(F1) **Machine Behavior** (3 features):
Consider Figure 4. Let S be the set of machines that query domain d, let $I \subseteq S$ be the subset of these machines that are known to be infected (i.e., are labeled as *malware*), and let $U \subseteq S$ be the subset of machines labeled as *unknown*. We measure three features: the fraction of known infected machines, $m = |I|/|S|$; the fraction of "unknown" machines, $u = |U|/|S|$; and the total number of machines, $t = |S|$, that query d. These features try to capture the fact that the larger the total number t and fraction m of infected machines that query d, the higher the probability that d is a malware-control domain.

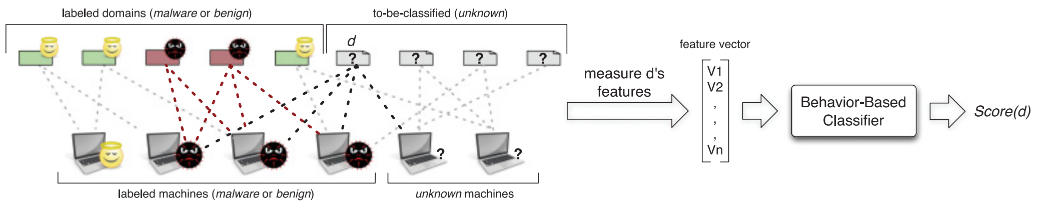

labeled domains (*malware* or *benign*)
to-be-classified (*unknown*)
d
?
labeled machines (*malware* or *benign*)
unknown machines
measure d's features
feature vector
V1
V2
Vn
Behavior-Based Classifier
Score(d)

Fig. 4. Overview of Segugio's feature measurement and classification phase. First, domain *d*'s features are measured, then the feature vector is assigned a "malware score" by the previously trained classifier.

(F2) **Domain Activity** (4 features):
Intuitively, newly seen domains are more likely to be malware related if they are queried mostly by known malware-infected machines. Registration information may be of help, but some malware domains may have a long registration period and remain "dormant" for some time, waiting to be used by the attackers. Instead of measuring the "age" of a domain, we aim to capture its *domain activity*. Let t_{now} be the day in which the graph $\mathcal{G}$ was built, and t_{past} be n days in the past with regard to t_{now} (e.g., we use $n = 14$ in our experiments). We measure the total number of days in which d was actively queried within the time window $[t_{now} - t_{past}]$, and the number of consecutive days ending with t_{now} in which d was queried. We similarly measure these two features for the effective second-level domain of d.

(F3) **IP Abuse** (4 features):
Let A be the set of IPs to which d resolved during our observation window T. We would like to know how many of these IPs have been pointed to in the past by already known malware-control domains. To this end, we leverage a large passive DNS database. We consider a time period W preceding t_{now} (e.g., we set $W = 5$ months in our experiments). We then measure the fraction of IPs in A that were associated to known malware domains during W. Also, for each IP in A, we consider its /24 prefix, and measure the fraction of such prefixes that match an IP that was pointed to by known malware domains during W. Similarly, we measure the number of IPs and /24s that were used by *unknown* domains during W.

Past Feature Use. It is worth noting that, while information similar to our *IP abuse* features (F3) has been used in previous work in different forms, for example, in Notos [Antonakakis et al. 2010], Exposure [Bilge et al. 2011], and [Kührer et al. 2014], we show in Section 5.4 that those features are helpful but not critical for Segugio to achieve high accuracy. In fact, the combination of our feature groups (F1) and (F2) by themselves already allows us to obtain quite accurate classification results. In addition, in Rahbarinia et al. [2015], we show that, by combining the *IP abuse* features with our *machine behavior* features, Segugio outperforms Notos.

Classifier Operation. To put Segugio in operation, we proceed as follows. Let $\mathcal{C}$ be Segugio's domain classifier trained during a traffic observation window T_1 (the training process is explained later in this section). Our main objective is to use $\mathcal{C}$ to classify *unknown* domains observed in DNS traffic from a different time window T_2. To this end, we first build a machine-domain graph $\mathcal{G}_{T_2}$ on traffic from T_2. Then, for each *unknown* (i.e., to be classified) domain $d \in \mathcal{G}_{T_2}$, we measure the statistical features defined earlier, as shown in Figure 4. Then, we input d's feature vector into the previously trained classifier $\mathcal{C}$, which computes a *malware score* for d. If this score is above a (tunable) detection threshold, we label d as *malware*. The detection threshold can be chosen to obtain the desired trade-off between true and false positives, which we evaluate in Section 5.

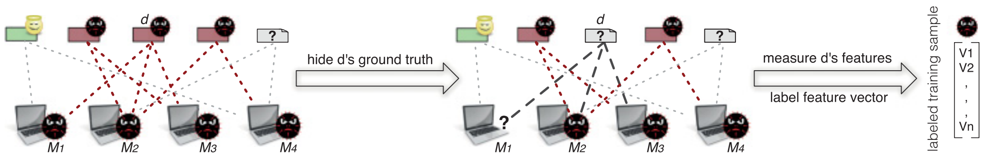


Fig. 5. Training set preparation: extracting the feature vector for a known malware-control domain. Notice that "hiding" d's label causes machine M_1 to also be labeled as *unknown* because, in this example, d was the only known malware-control domain queried by M_1. Machines M_2, M_3, M_4 queried some other known malware domains; therefore, they keep their original labels.

Training Dataset. To obtain the dataset used to train the classifier $\mathcal{C}$, we proceed as follows (see Figure 5). Let T_1 be the "training time" (e.g., one day). For each *benign* or *malware* domain d observed during T_1, we first temporarily "hide" its true label, then measure its features as defined earlier. The reason why we need to temporarily hide the ground truth related to d is precisely to enable feature measurement. In fact, our definition of features (delineated earlier) applies to *unknown* domains only, because if a domain is already known to be malware, its first two machine behavior features, for example, would be by definition always one and zero, respectively.

Note that hiding d's true label may have an impact on the label assigned to the machines that query it. For example, if d is a malware domain and there exists a machine that was labeled as malware *only* because it queried d, once we hide d's ground truth, that machine should also be relabeled as *unknown*, as shown for machine M_1 in the example in Figure 5. After measuring the features, we label the obtained feature vector with d's original label (see Figure 5). By repeating this process for every *malware* and *benign* domain, we obtain a dataset that can be used to train the statistical classifier $\mathcal{C}$ (e.g., using Random Forest [Breiman 2001], Logistic Regression [Fan et al. 2008], and so on).

3. EXPERIMENTAL SETUP

We deployed Segugio in three large regional ISP networks, one located in the Southeast, one on the Northwest Coast, and one in the western part of the United States. We refer to these ISP networks simply as ISP_1, ISP_2, and ISP_3. Note that this article is part of a study approved by an institutional review board; appropriate steps have been taken by our data provider to minimize privacy risks for the network users.

By inspecting the DNS traffic between the ISPs' customers and their local resolvers, we observed between roughly 1 to 4 million distinct machine identifiers per day (note that the identifiers that we were provided were stable, and did not appreciably suffer from DHCP effects). Most of our experiments with Segugio were conducted in the month of April, 2013. We randomly sampled four days of traffic from that month, per each of the ISP networks. Table I summarizes the number of distinct machines and domains observed in the traffic, and the (randomly) sampled days used in our evaluation.

Domain and Machine Labeling. To label the known *malware* domain names, we check if its entire domain name string matches a domain in our C&C blacklist. We made use of a large commercial C&C domain blacklist containing tens of thousands of recently discovered malware-control domains (in Section 5.6, we also report on experiments using public blacklists). The blacklist also has information related to the malware families of C&C domains. In total, the blacklisted C&C domains belong to about a thousand malware families. Note that malware families were determined based on labeling by a threat analysis group, and it is not purely based on malware family names. Rather, families are determined based on an identifier for the attacker (or group of attackers) that own and control the domains. The advantage of using a commercial blacklist is that domains are carefully vetted by expert threat analysts to

Table I. Experiment Data (Before Graph Pruning)

Traffic Source	Num. of Domains			Num. of Machines		Edges
	Total	Benign	Malware	Total	Malware	
ISP$_1$, Day 1 (Apr. 01)	~5.4M	~1.2M	12,120	~0.9M	32,327	~157.9M
ISP$_1$, Day 2 (Apr. 12)	~8.7M	~1.8M	28,158	~2M	72,583	~345.9M
ISP$_1$, Day 3 (Apr. 21)	~4.9M	~1.2M	13,005	~0.81M	27,907	~138M
ISP$_1$, Day 4 (Apr. 30)	~6M	~1.3M	14,385	~0.84M	29,803	~157.5M
ISP$_2$, Day 1 (Apr. 02)	~9M	~1.8M	13,239	~1.6M	50,339	~319.9M
ISP$_2$, Day 2 (Apr. 15)	~9M	~1.9M	20,277	~1.6M	49,944	~324.2M
ISP$_2$, Day 3 (Apr. 23)	~8.2M	~1.8M	18,020	~1.6M	47,506	~310.7M
ISP$_2$, Day 4 (Apr. 28)	~10M	~1.9M	11,597	~1.6M	44,299	~312.3M
ISP$_3$, Day 1 (Apr. 08)	~10.2M	~2M	15,706	~4M	78,990	~352.6M
ISP$_3$, Day 2 (Apr. 20)	~9.8M	~2M	14,279	~3.9M	74,098	~347.1M
ISP$_3$, Day 3 (Apr. 26)	~9.6M	~2M	36,758	~3.9M	69,773	~333.7M
ISP$_3$, Day 4 (Apr. 30)	~10.6M	~2.2M	13,467	~4M	72,519	~355.6M

minimize noise (i.e., mislabeled benign domains). All machines that query a known C&C domain are also labeled as *malware* because we assume that benign machines would have no reason to query "malware-only" C&C domains (see Section 7 for possible limitations).

To label known *benign* domains, we collected a 1-year archive of popular effective second-level domain (e2LD) rankings according to alexa.com. Specifically, every day for one year, we collected the list of top 1 million (1M) popular domain names. Then, we searched this large archive for domain names that consistently appeared in the top 1M list for the entire year. This produced a list of 458,564 popular e2LDs, which we used to label benign domains. Accordingly, we label a domain d as benign if its e2LD matches the whitelist. For example, we would label www.bbc.co.uk as benign, because its e2LD is bbc.co.uk , which is whitelisted.

The reason why we add only "consistently top" e2LDs to our whitelist is that sometimes malicious domains may become "popular" (due to a high number of victims) and enter the top 1M list for a brief period of time. The vast majority of such domains can be filtered out by the filtering strategy described earlier. In addition, we filter out e2LDs that allow for the "free registration" of subdomains, such as popular blog-publishing services or dynamic DNS domains (e.g., wordpress.com and dyndns.com), as their subdomains are often abused by attackers. At the same time, as mentioned in Section 2.1.1, we acknowledge that perfectly filtering all such "special" e2LDs may be difficult, and a small amount of noise may remain in the whitelist. In Section 5.6, we discuss how the possible remaining noise may potentially inflate the number of false positives that we measure. Note that whitelist noise may cause us to *underestimate* Segugio's true accuracy (i.e., the accuracy that we could otherwise achieve with a perfectly "clean" whitelist); we therefore believe that this is acceptable because it would *not* artificially favor our evaluation.

Table I summarizes the number of known benign and malware domains and machines that we observed; Figure 6 shows the cumulative distribution function for the number of machines that query the known malware domains. For example, we can see that 83.96% of known C&C domains are queried by less than 25 machines (we found these results to be consistent across different ISP networks and traffic observation days). This means that most malware-control domains are queried by a fairly low infected population. Nonetheless, in aggregate, the ISP networks host tens of thousands of infected machines.

Domain Node Annotations. For each day of traffic monitoring, we build a machine-domain bipartite graph, as discussed in Section 2.1. Each domain node

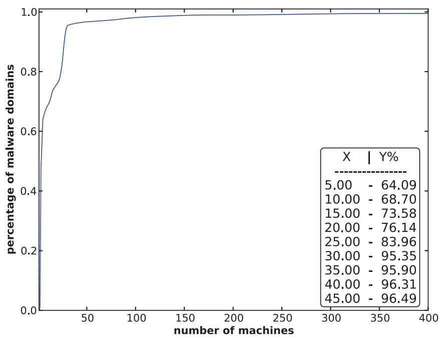


Fig. 6. Distribution of number of machines that query known malware C&C domains (measured from ISP_1, Day 1).

is augmented with information about the IP addresses that the domain resolved to during the observation day, and its estimated activity. Given a machine-domain graph built on a day t_i, to estimate the *domain activity* features (see Section 2.1.3) for a domain d, we consider DNS queries about d within 2 weeks preceding t_i. For estimating the resolved IP abuse features, we leverage a large passive DNS (pDNS) database, and consider pDNS data stored within 5 months before t_i.

Graph Pruning. Following the process described in Section 2.1.2, we prune the graph by applying our set of conservative filtering rules (R1 to R4). On average, the pruning process reduced the number of domain nodes by 26.55% and the machine nodes by 13.85%. Also, we obtained a 26.59% reduction in the total number of edges.

Definition of Domain TP and FP. In our experiments, we evaluate Segugio using TP and FP measures for domains. Domains from the blacklist that are correctly classified as malware-control are considered TP; FP domains are benign domains according to the whitelist that are incorrectly classified as malware-control domains.

4. SUMMARY OF RESULTS

To help the reader follow our evaluation, we briefly summarize the most important results and refer to the related sections for details:

—*Cross-validation tests* (Section 5.1): We performed extensive 10-fold cross-validation tests in three separate large ISP networks and on multiple different days of traffic. Segugio can consistently achieve a true-positive rate around 95% at 0.1% false positives.

—*Cross-day and cross-network tests* (Sections 5.2 and 5.3): Segugio can be trained on a given day of traffic from a given network, and can then be deployed in the same network or in a separate network to accurately detect new malware-control domains on traffic observed even several days later.

—*Feature analysis* (Section 5.4): We show that each of the three groups of features that we describe in Section 2.1 contributes meaningfully to Segugio's detection abilities, and at the same time that none of the feature groups by itself is absolutely critical to achieve high accuracy.

—*Cross-malware family tests* (Section 5.5): We show that Segugio can also discover domains related to machines infected with malware families previously unseen in the monitored networks, achieving 85% true positives at a false-positive rate of 0.1%.

—*Segugio's FP analysis* (Section 5.6): An analysis of possible causes of Segugio's false positives.

—*Additional results*: We provide the following additional results. Section 5.6: By relying on public blacklist information, we can still accurately detect new malware-control domains. Section 5.7: Segugio can detect new malware-control domains several days before they appear in malware C&C blacklists. Section 5.8: Our system can classify one entire day's worth of ISP-level DNS traffic in a few minutes.

—*Detection of new infected machines*: In Section 5.9, we show that the new malware-control domains discovered by Segugio can be successfully deployed in malware domain blacklists, enabling the detection of tens of thousands of previously unknown infected machines with low false positives.

—*Comparison with Belief Propagation*: In Section 6, we implemented the Loopy Belief Propagation algorithm [Koller and Friedman 2009] and compared Segugio's detection accuracy and efficiency with similar systems based on Belief Propagation, such as Manadhata et al. [2014].

5. EXPERIMENTAL RESULTS

5.1. Cross-Validation Tests

To evaluate Segugio's accuracy, we performed extensive 10-fold cross-validation experiments. Our main goal in setting up the experiments was to guarantee that no ground-truth information about the domains to be classified is ever used to measure the features or derive the training dataset. Namely, all training and test instances are derived from a machine-behavior graph whose nodes (both domains and machines) are labeled by pretending that all domains to be used for testing are *unknown*. To achieve this goal, we devised a rigorous procedure to build the training and test folds, as explained here.

Preparing the "10 folds." We now provide full details on the procedure that we use to set up the cross-validation experiments over a machine-domain bipartite graph. A summary of what we discuss here is given as an algorithm in Algorithm 1.

Let $\mathcal{G} = (M, D, E)$ be the pruned machine-domain graph derived from a given ISP network based on one day of traffic observation. Also, let D_m, D_b, and D_u be the subsets of domain nodes in D that are either known *malware*, *benign*, or *unknown*, respectively.

First, we take the labeled domain sets D_m and D_b, and randomly partition them into $k = 10$ different disjoint subsets. For example, we split the set of known malware-control domains D_m into k subsets $\{D_m^{(1)}, \ldots, D_m^{(k)}\}$. We do the same for the set of known benign domains, D_b. Then, we build k new machine-domain graphs $\{\mathcal{G}_1, \ldots, \mathcal{G}_k\}$, in which domain nodes of $\mathcal{G}_i$ that belong to the sets $D_m^{(i)}$ and $D_b^{(i)}$ are labeled as *unknown*. All other remaining domain nodes in $D_m^{(j)}$ and $D_b^{(j)}$, $\forall j \neq i$, are labeled according to their true labels, that is, as *malware* and *benign*, respectively. Finally, all domains in D_u remain labeled as *unknown*. In other words, graph $\mathcal{G}_i$ is built by "hiding" our ground truth about domains in $D_m^{(i)}$ and $D_b^{(i)}$, and labeling all domain nodes according to this new "partial" ground truth. We then label the machine nodes according to the new ground truth. For example, all machines that query any domain in $D_m^{(j)}$, $\forall j \neq i$, are labeled as *malware*, because, as mentioned earlier, we assume that machines that query known malware-control domains are infected. On the other hand, machines that queried one or more domains in $D_m^{(i)}$ but never queried any domain in $D_m^{(j)}$, $\forall j \neq i$, are labeled as *unknown*, because the ground truth associated to $D_m^{(i)}$ is "hidden" (it will be used only to compute the true positives during the test phase).

Feature measurement and test-set preparation. Now, let us consider one of these k graphs: say, $\mathcal{G}_i$. Our test data is represented by all nodes (i.e., all domains) in $D_m^{(i)}$ and $D_b^{(i)}$, whose true labels we want to recover by using our behavior-based classifier (remember that nodes in $\mathcal{G}_i$ are labeled after "hiding" the ground truth in $D_m^{(i)}$ and $D_b^{(i)}$).

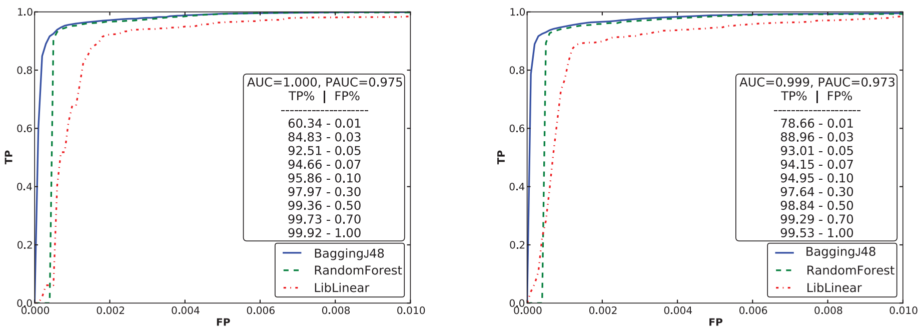
ALGORITHM 1: Graph-Based 10-Fold Cross-Validation

```
Data: $\mathcal{G} = (M, D, E)$
$D_m$: known malware domains
$D_b$: known benign domains
$D_u$: unknown domains
Result: 10-fold cross-validation results
$k = 10$;
split $D_m$ into $k$ disjoint sets $\{D_m^{(1)}, \dots, D_m^{(k)}\}$ s.t. $D_m = \bigcup_{i=1}^{k} D_m^{(i)}$;
split $D_b$ into $k$ disjoint sets $\{D_b^{(1)}, \dots, D_b^{(k)}\}$ s.t. $D_b = \bigcup_{i=1}^{k} D_b^{(i)}$;
for $i \leftarrow 1$ to $k$ do
    build $\mathcal{G}_i = (M_i, D_i, E_i)$ by hiding the ground truth for $D_m^{(i)}$ and $D_b^{(i)}$;
    relabel $M_i$ according to the new (partial) domain ground truth;
    $V_m^{(i)} = \{$set of feature vectors $\forall d \in D_m^{(i)}\}$;
    $V_b^{(i)} = \{$set of feature vectors $\forall d \in D_b^{(i)}\}$;
    $V^{(i)} = V_m^{(i)} \cup V_b^{(i)}$ ($i$th fold test data);
    $W_m^{(i)} = \{$set of feature vectors $\forall d \in D_m^{(\neg i)} = \bigcup_{j \neq i} D_m^{(j)}$ computed by temporarily hiding $d$'s true
    label$\}$;
    $W_b^{(i)} = \{$set of feature vectors $\forall d \in D_b^{(\neg i)} = \bigcup_{j \neq i} D_b^{(j)}$ computed by temporarily hiding $d$'s true
    label$\}$;
    $W^{(i)} = W_m^{(i)} \cup W_b^{(i)}$ ($i$th fold training data);
    $\mathcal{C}^{(i)} \leftarrow$ train a classifier on $W^{(i)}$;
    $P^{(i)} \leftarrow$ classify feature vectors in the test set $V^{(i)}$ using $\mathcal{C}^{(i)}$ and store their malware scores;
end
$(TPs, FPs) \leftarrow$ set the detection threshold based on all scores in $P = \bigcup_{i=1}^{k} P^{(i)}$ to obtain the
desired trade-off between true and false positives;
return $P$ and $(TPs, FPs)$
```

To this end, for each domain node in $D_m^{(i)}$ and $D_b^{(i)}$, we measure the statistical features as defined in Section 2.1 (see also Figure 4). Let $V_m^{(i)}$ and $V_b^{(i)}$ be the set of all feature vectors derived from the domains in $D_m^{(i)}$ and $D_b^{(i)}$, respectively. While for the purpose of feature measurement we pretend that the domains are *unknown*, we do know their true labels; therefore, we can eventually label their vectors, thus obtaining the desired labeled test set $V^{(i)} = V_m^{(i)} \cup V_b^{(i)}$. By applying this process for each $i = 1, \dots, k$, we obtained the required $k = 10$ different test datasets, one per each fold.

Training set preparation. To prepare the training dataset for the ith cross-validation fold, we proceed as follows. We consider the pruned graph $\mathcal{G}_i$, and the set of its labeled domain nodes $D_m^{(\neg i)} = \bigcup_{j \neq i} D_m^{(j)}$ and $D_b^{(\neg i)} = \bigcup_{j \neq i} D_b^{(j)}$, namely, all *malware* and *benign* domains excluding the "test domains" in $D_m^{(i)}$ and $D_b^{(i)}$ (which, as explained earlier, we pretend are labeled as *unknown*). Then, we apply the training dataset preparation procedure explained in Section 2.1.3 (see also Figure 5), thus obtaining the ith fold's training dataset.

10-fold cross-validation results. Given one day of traffic from an ISP network, we prepare the training and test sets as explained earlier, then apply the standard cross-validation procedure to evaluate our classifier. For each fold, we compute the ROC curve by varying the detection threshold on the classifier's output scores, the area under the curve (AUC), and the *partial* AUC (PAUC). Note that the PAUC is computed by measuring the area under the ROC curve in a range of false positives between 0%



(a) ISP_1 (b) ISP_3

Fig. 7. Cross-validation results for two different ISP networks (with one day of traffic observation; FPs in $[0, 0.01]$).

and 1%, and by normalizing this measure to obtain a value in $[0, 1]$. In essence, the PAUC highlights the classifier's trade-off between TPs and FPs at very low FP rates. We then average these results across all folds.

Figure 7 shows results obtained for ISP_1 and ISP_3 networks (the ISP_2 result is very similar and is omitted due to space constraints). We conducted our experiments by building the machine-domain graphs based on one day of DNS traffic, and by comparing three different learning algorithms: a multiple-classifier system built using bagged decision trees [Kuncheva 2004] (BaggingJ48), the Random Forest algorithm by Breiman [2001] (RandomForest), and an efficient implementation of Logistic Regression with $\mathcal{L}_1$ regularization [Fan et al. 2008] (LibLinear). As shown in Figure 7, we consistently obtained the best results with BaggingJ48, achieving an AUC close to 100% and a PAUC above 97.5%. Also, the TP-FP trade-off tables embedded in the graphs show that we can achieve above 95% true positives at a false-positive rate of 0.1%. These results are consistent across all ISP networks and days of traffic that we had available. Our detailed analysis of FPs is presented in Section 5.6.

5.2. Cross-Day Tests

In this section, we aim to show that a behavioral-based classifier learned on a given day can be successfully deployed and can accurately classify new malware-control domains observed several days after the training was completed. To this end, we perform a large number of cross-day train-test experiments. In the following, we first explain how we prepared the training and test datasets used in this set of experiments, then report a representative set of the results that we obtained.

Preparation of the datasets. To prepare the training and test sets, we consider two days of traffic at a given ISP network. These two days could be consecutive days or they could be separated by a "gap" of several days. We use the DNS traffic from the first day for training purposes, then test our Segugio classifier on the second day of traffic. We devised a rigorous procedure to make sure that no ground truth information about the test domains is ever used during training and feature measurement. The following explains the full details on how the training and test datasets are constructed.

Training set preparation. To prepare the training set, we first built the machine-domain graphs $\mathcal{G}_{t_1}$ and $\mathcal{G}_{t_2}$ according to the DNS traffic observed at a given ISP network on two different days, t_1 and t_2, respectively. These two days of traffic do not need to be consecutive; in fact, in our experiments, they are separated by a gap of several days.

Our main goal in preparing the training set was to make sure that a large subset of the known malware and benign domains that appear in day t_1 and day t_2 are *excluded* from training, and are *used only for testing*. This allowed us to evaluate the classifier's generalization ability, and how accurately we can detect previously unknown malware domains. To achieve this goal, we proceeded as follows. Let $D_m^{t_1}$ and $D_m^{t_2}$ be the sets of known malware-control domains that were "visible" in the traffic (i.e., that were queried at least once) on day one and day two, respectively. Similarly, let $D_b^{t_1}$ and $D_b^{t_2}$ be the sets of known benign domains observed in the traffic on these two different days. First, we computed the set intersections $D_m^{t_{12}} = D_m^{t_1} \cap D_m^{t_2}$, and $D_b^{t_{12}} = D_b^{t_1} \cap D_b^{t_2}$. In other words, we find all known malware and benign domains that were queried both in day one and day two. Then, we randomly "hide" the ground truth for half of the malware-control domains in $D_m^{t_{12}}$, effectively relabeling them as *unknown*. We do the same for the benign domains in $D_b^{t_{12}}$, and call $D_m^{t_{12;50\%}}$ and $D_b^{t_{12;50\%}}$ the resulting sets of domains whose ground truth was "hidden.' The reason why we "hide" the true label of domains in these two sets is that we will later use them for test purposes. Therefore, we need to effectively remove them from the training set of malware and benign domains.

Let $\mathcal{G}_{t_{1;50\%}}$ be the graph $\mathcal{G}_{t_1}$ relabeled by excluding the ground truth for domains in $D_m^{t_{12;50\%}}$ and $D_b^{t_{12;50\%}}$ (i.e., those domain nodes are relabeled as *unknown*, and the machines that query them are relabeled accordingly). For each remaining labeled domain node in this new graph, we measure its features as explained in Section 2.1 (see also Figure 5), thus obtaining our training dataset.

Test set preparation. To prepare the test set, we consider graph $\mathcal{G}_{t_2}$, which was built on day two. Let $N_m = D_m^{t_2} - D_m^{t_{12}}$ be the set of "new" known malware domains that we observed in the traffic of day t_2, but not in t_1 (i.e., those domains that were queried on day two but not on day one). Similarly, let $N_b = D_b^{t_2} - D_b^{t_{12}}$ be the set of "new" benign domains. We first randomly split N_m and N_b in half, obtaining two sets $N_m^{50\%}$ and $N_b^{50\%}$ by picking one of the halves per set. Then, we "hide" the true labels of domains in these two sets, effectively pretending that they are labeled as *unknown*.

Now, we form our test dataset of malware domains as $T_m = N_m^{50\%} \cup D_m^{t_{12;50\%}}$, and $T_b = N_b^{50\%} \cup D_b^{t_{12;50\%}}$ for the benign domains. Then, given graph $\mathcal{G}_{t_2}$, we relabel it according to all the available ground truth, excluding all domains in T_m and T_b, thus obtaining a new labeled graph. Finally, we use this new (partially labeled) graph to measure the features of each test domain in T_m and T_b, following the process described in Section 2.1 (see also Figure 4).

Ultimately, the experimental approach outlined here allows us to obtain a large labeled test dataset of domains that were never used for training and whose true labels are never used during the feature measurement process. Figure 8 schematically shows how the aforementioned sets of domains for training and testing are generated for malware-control domains (diagram of benign domains would be symmetrical). As can be seen from the diagram, no information from the domains used for testing (red and orange shadings) was ever used for training. The number of test samples we obtained this way are reported in Table II.

Cross-day test results. We used multiple training and test sets to evaluate our behavior-based classifier on all three ISP networks, and on several combinations of dates for day t_1 and t_2. The TP rate is computed by dividing the number of correctly classified malicious test domains by the total number of malicious domains in the same test dataset (e.g., 6,767 for the ISP_3 experiments in Table II). The FP rate is computed in a similar way by considering the benign test domains. The classification results for ISP_1 and ISP_3 experiments are reported in Figure 9 (ISP_2 results are reported in the conference version of this article [Rahbarinia et al. 2015]). Segugio was able to consistently achieve above 93% TPs at 0.1% FPs.

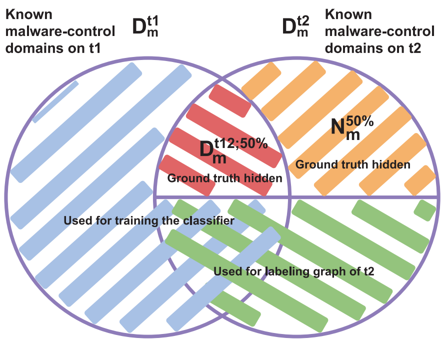
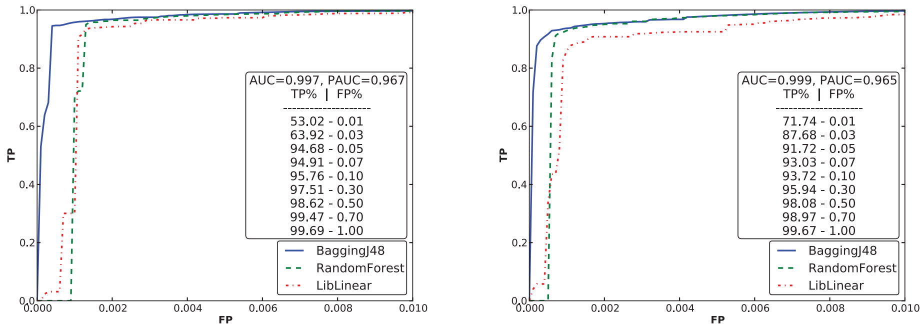

Known malware-control domains on t1
D_m^{t1}
D_m^{t2}
Known malware-control domains on t2
$D_m^{t12;50\%}$
Ground truth hidden
$N_m^{50\%}$
Ground truth hidden
Used for training the classifier
Used for labeling graph of t2

Fig. 8. Depiction of the generation of train and test datasets for malware-control domains.

Table II. Cross-Day Test Set Sizes

Experiment	T_m	T_b	$D_m^{t12;50\%}$	$D_b^{t12;50\%}$	$N_m^{50\%}$	$N_b^{50\%}$
ISP_1	5,819	514,053	2,000	304,127	3,819	209,926
ISP_2	9,980	780,707	2,054	432,731	7,926	347,976
ISP_3	6,767	816,666	2,076	434,509	4,691	382,157

Note: Includes the total size and the size of the test subsets.



AUC=0.997, PAUC=0.967

TP%	FP%
53.02	0.01
63.92	0.03
94.68	0.05
94.91	0.07
95.76	0.10
97.51	0.30
98.62	0.50
99.47	0.70
99.69	1.00

AUC=0.999, PAUC=0.965

TP%	FP%
71.74	0.01
87.68	0.03
91.72	0.05
93.03	0.07
93.72	0.10
95.94	0.30
98.08	0.50
98.97	0.70
99.67	1.00

(a) ISP_1 (train-test gap: 11 days) (b) ISP_3 (train-test gap: 12 days)

Fig. 9. Cross-day test results for two ISP networks (FPs in [0, 0.01]).

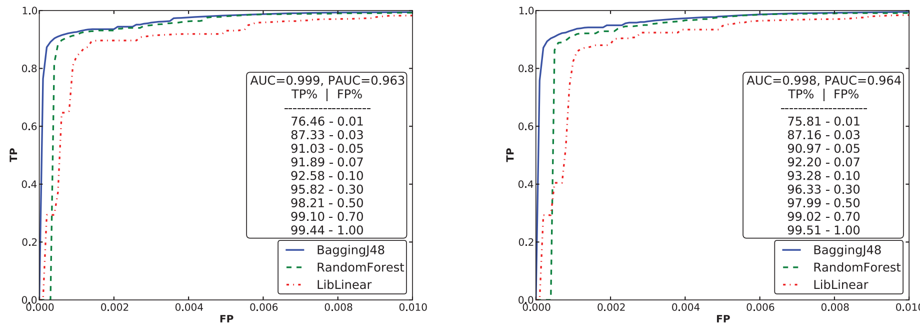
To choose the best detection threshold for real-world deployment, one could perform pilot experiments to compute TP and FP rates and then choose the desired operation point on the ROC curve.

5.3. Cross-Network Tests

To show that our behavior-based classifier can perform well when trained in an ISP network and deployed in another, we performed extensive "cross-network" tests. This set of experiments is conducted in a way very similar to the cross-day tests described in Section 5.2. The only difference is that the second day of traffic, t_2, comes from a network that is different from the one in which the day t_1 traffic was observed. All other details are the same. That is, we prepare the training and test sets following the same approach that we described in Section 5.2.

Figure 10 shows our results for the following two cases: (a) train on a given day of traffic from ISP_1, and test on a different day of traffic observed at ISP_3's network; and (b) train on a given day of traffic from ISP_2, and test on a different day of traffic observed at ISP_3's network. As we can see, even when deployed on a different network, Segugio can achieve more than 92% TP for an FP rate of 0.1%. We provide detailed qualitative analysis of FP domains in Section 5.6.

We experimented with various combinations of networks and test days. Two of these combinations are shown in Figure 10. Other combination results are very similar to the



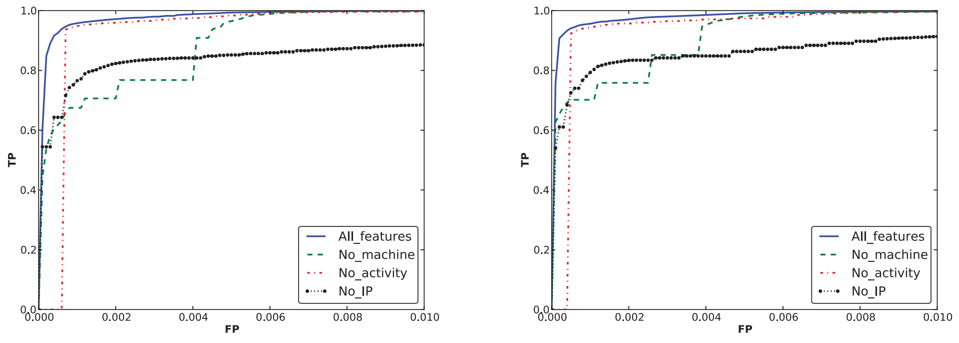
(a) Train on ISP_1, test on ISP_3 (gap: 14 days). (b) Train on ISP_2, test on ISP_3 (gap: 15 days).

Fig. 10. Cross-network test results for two different ISP networks (FPs in $[0, 0.01]$).

Table III. Cross-Network Test Sets Size

Experiment	T_m	T_b	$D_m^{t12;50\%}$	$D_b^{t12;50\%}$	$N_m^{50\%}$	$N_b^{50\%}$
ISP_1,ISP_3	6,476	879,328	1,902	317,206	4,574	562,122
ISP_2,ISP_3	6,477	879,328	2,165	406,558	4,312	472,770

Note: Includes total size and size of the test subsets.



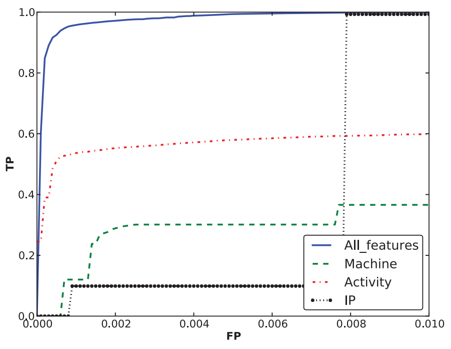
(a) ISP_1 (b) ISP_3

Fig. 11. Feature analysis: results obtained by excluding one group of features at a time, and comparison to using all features (FPs in $[0, 0.01]$).

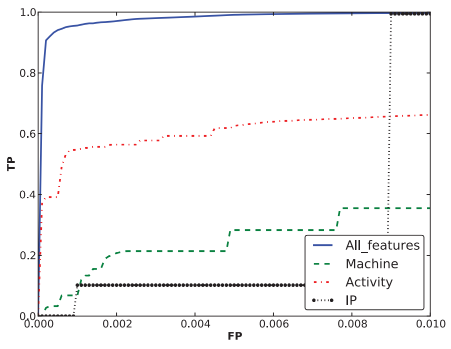
ones reported and could be found in the conference version of this article [Rahbarinia et al. 2015]. Table III reports the number of test samples for each of the two experiments that we report.

5.4. Feature Analysis

We also performed a detailed analysis of our statistical features by training and testing Segugio after completely *removing one of the three feature groups* described in Section 2.1.3 at a time (using BaggingJ48). For example, in Figure 11, the "No IP" ROC curves (dashed black line) refer to a statistical classifier learned without making use of the *IP abuse* features (F3). As we can see, even without the IP abuse features, Segugio can consistently achieve more than 80% TPs at less than 0.2% FPs. Also, we can see from the "No machine" line that removing our *machine behavior* (F1) features (i.e., using only domain activity and IP abuse features) would cause a noticeable drop in the TP rate for most FP rates below 0.5%. This shows that our machine behavior



(a) ISP_1



(b) ISP_3

Fig. 12. Feature analysis: results obtained by using only one group of features at a time, and comparison with results obtained using all features (FPs in $[0, 0.01]$).

features are needed to achieve high detection rates at low false positives. Overall, the combination of all three feature groups yields the best results.

Figure 12 shows the ROC curves obtained by *using only one group of features* at a time. The figure shows that at low FPs, the IP abuse features by themselves yield a worse TP rate than using the *machine behavior* or *domain activity* features (see the "IP" ROCs, which refer to cross-validation results obtained with a classifier trained using *only* the IP abuse features). For the sake of brevity, Figure 12 reports only the results obtained at ISP_1 and ISP_3. The results obtained at ISP_2 are similar. Note the spikes in Figure 12 for the IP curve (dotted black line), for example. These spikes occur when the IP abuse feature values (ratio of historically known bad IPs to all IPs for domains) increase to a point that cause the prediction scores produced by the classifier to exceed a certain detection threshold. This enables the classifier to make a good distinction between malware-control domains and benign domains.

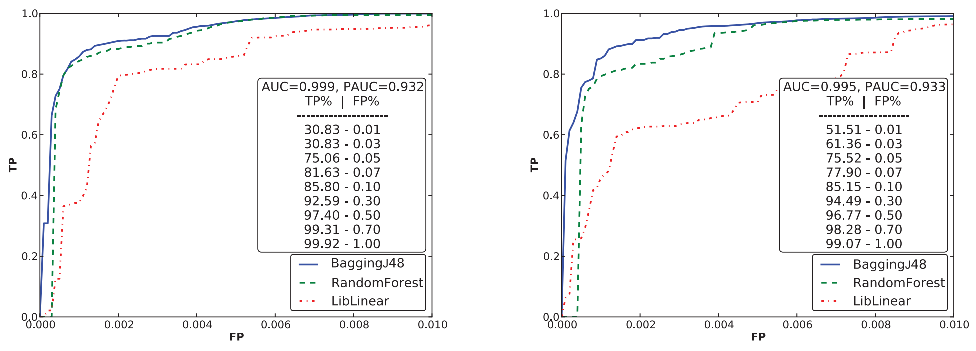
5.5. Cross-Malware Family Tests

While Segugio's main goal is to discover the occurrence of new malware-control domains by tracking known infections, in this section, we show that Segugio can also detect domains related to malware families previously unseen in the monitored networks. Namely, no infection related to those families was previously known to have occurred in the monitored networks.

To this end, we performed a set of experiments by splitting our dataset of known blacklisted C&C domains *according to their malware family*, rather than at random. The source of our commercial blacklist was able to provide us with malware family labels[3] for the vast majority of blacklisted domains (less than 0.1% of blacklisted domains were excluded form these experiments). Overall, the blacklist consisted of tens of thousands of C&C domains divided into more than 1,000 different malware families.

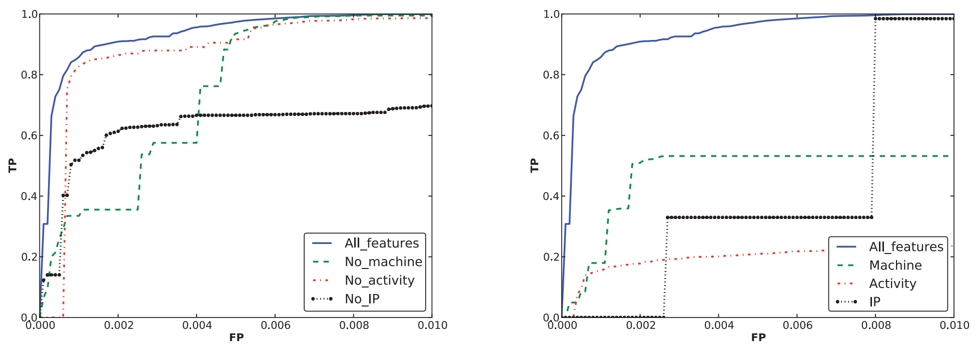
To prepare our new tests, we devised an approach similar to standard cross-validation and partitioned the blacklisted domains into *balanced sets (or folds) of malware families*. Namely, each fold contained roughly the same number of malware families. The net result is that the domains used for testing always belonged to malware families

[3]Often, the labels were more fine-grained than generic malware families, and associated domains to a specific cyber-criminal group.



(a) ISP_1 (b) ISP_3

Fig. 13. Cross-malware family results (one day of traffic observation from ISP_1 and ISP_3; FPs in [0, 0.01]).



(a) ISP_1 (b) ISP_1

Fig. 14. Feature analysis for cross-malware family tests. (a) Results are obtained by excluding one group of features at a time. The graph also reports the results obtained using all features, for comparison. (b) Results obtained using only one group of features at a time (FPs are in [0, 0.01]).

never used for training. Said another way, *none of the known malware-control domains used for training belonged to any of the malware families represented in the test set*.

The results are reported in Figure 13 (due to space limitations, we present results for ISP_1 and ISP_3 only; results for ISP_2 are very similar). As we can see, Segugio is able to discover domains related to new malware families with more than 85% TPs at 0.1% FPs. To explain this result, we performed a set of feature analysis experiments (similar to Section 5.4 and with BaggingJ48 as the classifier) using the new experiment settings. Using ISP_1 as an example and from Figure 14(a), we can see that if we remove the (F1) group of *machine-behavior* features, the detection rate drops significantly. In other words, our machine-behavior features are important, because using only feature groups (F2) and (F3) yields significantly lower detection results for low FP rates.

In addition, Figure 14(a) shows that, for this task of detecting new malware families, the (F3) *IP abuse* features are also important ("No IP" line), though the combination of feature groups (F1) and (F2) by themselves can still yield more than 50% new malware family detection at less than 0.2% FPs. On the other hand, removing the (F2) *domain-activity* features has a somewhat smaller impact on detection performance. Figure 14(b) reports similar feature analysis results related to including only one group of features at a time, rather than excluding one feature group at a time.

thaisqz.sites.**uol.com.br**
jkishii.sites.**uol.com.br**
sjhsjh333.**egloos.com**
ivoryzwei.**egloos.com**
dat007.**xtgem.com**
vk144.**narod.ru**
jhooli10.**freehostia.com**
7171.**freehostia.com**
cr0s.**interfree.it**
cr0k.**interfree.it**
id11870.**luxup.ru**
id23166.**luxup.ru**
...

Fig. 15. Example set of domains that were counted as false positives. The effective 2LDs are highlighted in bold.

One reason for the contribution of our machine-behavior features (F1) is the existence of *multiple infections*. Some machines appear to be infected with multiple malware belonging to different families, possibly due to the same vulnerabilities being exploited by different attackers, to the presence of malware droppers that sell their infection services to more than one criminal group, or because of multiple infections behind a NAT device (e.g., in the case of home networks). This observation has also been shown in some recent studies (Rossow et al. [2013] and Caballero et al. [2011]). Also, the domain activity features (F2) may help because the new domains were only recently used. Finally, the IP abuse features (F3) may help when new malware families point their control domains to IP space that was previously abused by different malware operators (e.g., in the case of the same bulletproof hosting services used by multiple malware owners).

5.6. Analysis of Segugio's False Positives

We now provide an analysis of domains in our top Alexa whitelist that were classified as *malware* by Segugio. It is worth remembering that the whitelist that we use contains only effective second-level domains (e2LDs) that have been in the top 1 million list for an entire year (see Section 3 for more details). During testing, we count as false positive any fully qualified domain (FQD) classified by Segugio as *malware* whose e2LD is in our whitelist.

By analyzing Segugio's output, we found that most of the false positives are due to domains related to personal websites or blogs with names under an e2LD that we failed to identify as offering "free registration" of subdomains. As discussed in Section 3, such e2LDs may introduce noise in our whitelist, and should have been filtered out. For example, most of Segugio's false positives were related to domain names under e2LDs such as egloos.com, freehostia.com, uol.com.br, interfree.it, and so on. These types of services are easily abused by attackers. Consequently, many of the domains that we counted as false positives may very well be actual malware-control domains. Figure 15 shows an example subset of such domains.

We now provide a breakdown of the false positives generated by Segugio during three different cross-day and cross-network tests reported in Sections 5.2 and 5.3. Table IV summarizes the results. For example, experiment (a) produced 724 distinct false-positive FQDs, using a detection threshold set to produce at most 0.05% FPs and >90% TPs. Many of these FP domains shared the same e2LD. In fact, we had only 401 distinct e2LDs. Of these, the top 10 e2LDs that contributed the most FQDs under their domain name caused 32% of all FPs.

Table IV. Analysis of Segugio's FPs

Test Experiment	(a) ISP_2 cross-day	(b) ISP_3 cross-day	(c) ISP_2-ISP_3 cross-network
Absolute number of false positives for overall 0.05% FPs and > 90% TPs			
Fully qualified domains (FQDs)	724	807	786
Effective second-level domains (e2LDs)	401	410	451
Contribution of top 10 e2LDs	230 (32%)	308 (38%)	247 (31%)
Featurecontributions			
$>$90% infected machines	73%	71%	55%
Past abused IPs	86%	85%	80%
Active for $\leq$ 3 days	26%	20%	27%
Evidence of malware communications (sandbox traces)			
Domains queried by malware	21%	23%	19%

Table IV also shows that 73% of all false-positive domains were queried by a group of machines, more than 90% of which were known to be infected. Also, 86% of the FP domains resolved to previously abused IP addresses, and 26% were active for only less than three days. Finally, using a separate large database of malware network traces obtained by executing malware samples in a sandbox, we found that 21% of the domains that we counted toward the false positives had been contacted by known malware samples. Note that these observations do not necessarily mean that the FP domains are indeed malware-control domains. For example, a malware could contact benign websites to check its Internet connectivity or for other reasons. However, since Segugio at the same time also labeled these domains as malware-control, this may indicate that a few could, in fact, be malware-control domains that are incorrectly present in our whitelist.

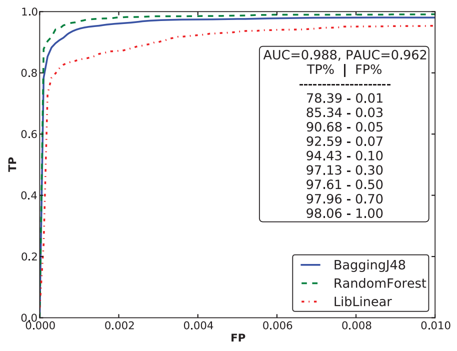
To summarize, our experiments show that Segugio's false-positive rate is low (e.g., $\leq$0.05% FPs at a TP rate $\geq$90%) and FPs may also be somewhat overestimated. In general, Segugio yields much lower FPs than previously proposed systems for detecting malicious domains (see Rahbarinia et al. [2015] for a comparison to Notos [Antonakakis et al. 2010]). Even so, we acknowledge that some false positives are essentially inevitable for statistical detection systems such as Segugio. Therefore, care should be taken (e.g., via an additional vetting process) before the discovered domains are deployed to block malware-control communications.

To show that Segugio's results are not critically dependent on the specific commercial malware C&C blacklist that we used as our ground truth, we also performed a number of experiments using public blacklist information.

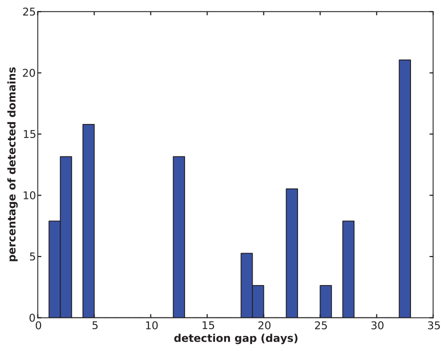
Cross-Day Tests. We repeated the cross-day experiment on machine-domain graphs labeled using exclusively known malware-control domains collected from public blacklists. More specifically, we collected domains labeled as malware C&C (we excluded other types of non-C&C malicious domains) from the following sources: spyeyetracker.abuse.ch, zeustracker.abuse.ch, malwaredomains.com, and malwaredomainlist.com. Overall, our public C&C domain blacklist consisted of 4,125 distinct domain names. We then used this blacklist to label the *malware* nodes in the machine-domain graph, then performed all other steps to conduct cross-day experiments using the same procedure described in Section 5.2 (the only change was the blacklist).

Figure 16(a) reports the results on traffic from ISP_2 (results for the other ISP network and different days of traffic are very similar). Segugio was able to achieve over 94% true positives at a false-positive rate of 0.1%.

Cross-Blacklist Tests. To further demonstrate Segugio's ability to discover new malware-control domains, we conducted another experiment by using our commercial C&C blacklist (described in Section 3) for training purposes, then testing Segugio to



(a)



(b)

Fig. 16. (a) Cross-day results using only public blacklists; (b) Early detection results: histogram of the time gap between Segugio's discovery of new malware-control domains and the time when they first appeared on the blacklist.

see if it would be able to detect new malware-control domains that appeared in the public blacklists but were not in our commercial blacklist (therefore were not used during training). By inspecting a day of traffic from ISP_2, we observed 260 malware-control domains that matched our public blacklist. However, of these 260 domains, 207 domains already existed in our commercial blacklist as well. Therefore, we used only the remaining 53 new domains that matched the public blacklist (but not the commercial blacklist) to compute Segugio's true positives. We found that Segugio could achieve the following trade-offs between true and false positives: TPs=57%, FPs=0.1%; TPs=74%, FPs=0.5%; and TPs=77%, FPs=0.9%. While the TP rate looks somewhat lower than what was obtained in other tests (though still fairly good, considering the low FP rates), we believe that this is mainly due to the limited test set size (only 53 domains) and noise. In fact, we manually found that the public blacklists that we used contained a number of domains labeled as C&C that were highly likely benign (e.g., recsports.uga.edu and www.hdblog.it), and others that were likely not related to malware-control activities (though possibly used for different malicious activities), which would not be labeled as *malware* by Segugio.

5.7. Early Detection of Malware-Control Domains

We also performed experiments to measure how early Segugio can detect malware-control domains compared to malware domain blacklists. To this end, we selected four consecutive days of data from each of the three ISP networks (12 days of traffic, overall). For each day, we trained Segugio and set the detection threshold to obtain $\leq$0.1% FPs. We then tested the classifier on all domains that on that day were still labeled as *unknown*. Finally, we checked if the new malware-control domains that we detected appeared in our blacklists in the following 35 days. During the four days of monitoring, we found 38 domains that later appeared in the blacklist. A large fraction of these newly discovered domains were added to the blacklist many days after they were detected by Segugio, as shown in Figure 16(b). Additionally, we measured how many domains were added to the blacklist but Segugio did not label them as malware-control. For example, when Segugio is tuned to produce at most 1% FPs, it missed only 2 domains from the traffic that were eventually included in the blacklist.

5.8. Segugio's Performance (Efficiency)

Segugio is able to efficiently learn the behavior-based classifier from an entire day of ISP-level DNS traffic, and can classify all (yet unknown) domains seen in a network in a matter of a few minutes. To show this, we computed the average training and test time for Segugio across the 12 days of traffic used to perform the *early-detection* experiments discussed in Section 5.7. On average, the learning phase took about 60 minutes, for building the graph, annotating and labeling the nodes, pruning the graph, and training the behavior-based classifier. The feature measurement and testing of all *unknown* domains required only about 3 minutes.

5.9. Detecting Previously Unknown Infected Machines

The experiments that we discuss in this section aim at evaluating the impact that Segugio may have on detecting previously unknown malware-infected machines. To this end, we performed the following experiments. We first performed a cross-day test, in a way similar to what we described in Section 5.2, using the traffic from two different days, t_1 and t_2. Remember that, during a cross-day test, we "hide" the ground truth of a large subset of known malware and benign domains. We then classified these domains using Segugio, and computed the true-positive and false-positive domains.

Let D_{tp} be the set of domains that were correctly classified as *malware* by Segugio, and D_{fp} be the set of benign domains that were misclassified as malware related. We take all domains in $B = D_{tp} \cup D_{fp}$ and create a blacklist, which we "deploy" on a separate later day, t_3, of traffic. Note that the blacklist B includes the false-positive domains, because they were classified as *malware* by Segugio, and in a real deployment we would not know that they are false positives. We then count how many machines query any domain $d \in B$ during t_3, and we check how many of these machines were actually known infected machines and how many were *benign* or *unknown* (i.e., we did not have evidence of infection). If a known infected machine queries a domain in B, we consider it to be a *true-positive* machine. On the other hand, if a *benign* or *unknown* machine queries a domain in B, we count it toward the *false-positive* machines.

Table V reports the percentage of TP and FP domains output by Segugio on day t_2 (i.e., as a result of the cross-day test) for different values of Segugio's malware detection threshold, and the percentage and absolute number of TP and FP machines classified on day t_3. Looking at the second line of results, we can see that Segugio correctly detected 95.6% of all the "hidden" malware domains, and misclassified only 0.05% of the benign domains. Matching these domains on t_3 allowed us to "recover" (i.e., correctly classify) 97.1% of malware-infected machines (i.e., 14,350 machines), with only 91 false positives, which account for only 0.003% of what we considered as noninfected machines, that is, machines whose original labeled was either benign or *unknown*. These experiments were conducted in ISP_3, with almost 3 million machine nodes in the pruned graph on which the tests are based.

5.10. Longitudinal Measurement of Detection Accuracy

In Section 5.3, we showed that Segugio's detection accuracy does not change when a trained model is taken to another ISP network and with a gap of several days. In this section, we conduct an experiment to provide a complete picture on how detection accuracy changes over time and to address how often Segugio needs to be retrained. Figure 17 shows TP rates obtained over a period of 1 month without retraining the classifiers (FP rate is fixed at 0.5%). Specifically, in this experiment, we first trained Segugio's classifier, then deployed it on multiple days in the following 30 days after the initial training. Note that Segugio was not retrained in this period of 30 days. As can be seen from Figure 17, Segugio's performance starts to deteriorate after about 15 days. This suggests that, to maintain high accuracy, Segugio should be retrained periodically. It is worth noting that the cost of retraining Segugio is not high, as it could be done in a matter of minutes (refer to Section 5.8).

Table V. Detection of Infected Machines

Total machines	Domains		Machines	
	TP %	FP %	TP %	FP %
∼3M ; ISP_3	88.82	0.01	6.5 (957)	0.000 (1)
∼3M ; ISP_3	95.60	0.05	97.1 (14350)	0.003 (91)
∼3M ; ISP_3	97.22	0.10	98.0 (14483)	0.009 (265)
∼3M ; ISP_3	99.32	0.50	99.0 (14627)	0.062 (1775)
∼3M ; ISP_3	99.59	0.70	99.1 (14641)	0.077 (2229)

Note: In parentheses, we report the absolute number of machines.

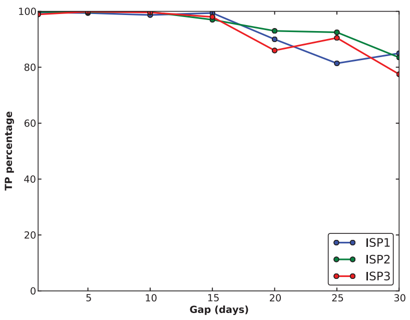


Fig. 17. Detection accuracy over 1-month period without retraining Segugio.

6. COMPARISON WITH BELIEF PROPAGATION

In this section, we compare Segugio to a detection system based on Belief Propagation (BP) and inspired by Manadhata et al. [2014] and Chau et al. [2011]. To this end, we treat each domain node in the bipartite graph $\mathcal{G}$ as a random variable X, which takes values in the set $\{good, bad\}$. A realization $X_i = good$ signifies that the ith domain is not a malware-control domain, whereas $X_i = bad$ would signify that the domain is a malware-control domain. Each domain has a probability of being *good* or *bad*, which is what we ultimately want to estimate. In other words, the ultimate goal is to compute the probability $P(X_i = good)$ or $P(X_i = bad)$. However, given a very large number of domains, answering this query efficiently is challenging. To solve this problem, we implement a distributed Loopy Belief Propagation [Koller and Friedman 2009] algorithm on GraphLab [Low et al. 2010]. Specifically, we model this problem using *conditional random fields*, and compute an approximate estimate of $P(X_i = bad)$.

The BP algorithm will output a probability for each domain node based on prior knowledge about the domain and the neighbors of the domain in the graph (i.e., the machines). Before running the Loopy BP algorithm on our graph, we need to assign *node potentials* to all nodes in our bipartite graph, including the domains and the machines. Node potentials express our *a priori* knowledge about the true nature of the domains. Also, every edge in the graph needs to be assigned an *edge potential*. Edge potentials represent the "compatibility" between the true nature of a domain d_i and a machine m_j that queried it, and vice versa. In our experiments, we employ two different standard approaches to assign the node and edge potentials. We compare and contrast these two methods with each other and with Segugio, and show that Segugio outperforms the detection results obtained via the BP algorithm.

The first way to assign the potentials is based on our prior knowledge about the nodes and the intuition about the "compatibility" of nodes that are connected to each other by an edge. Nodes that have a known label are assigned a node potential of $P(X_i = bad) = 0 + \epsilon$ if they are known "good" nodes (i.e., benign domains), and a node potential of $P(X_i = bad) = 1 - \epsilon$ if they are known "bad" nodes (i.e., malware-control domains and infected machines), where ϵ is a small constant. All other unknown nodes are assigned a neutral potential of $P(X_i = bad) = 0.5$. The edge potentials are assigned in a way similar to Chau et al. [2011]. In essence, this method of assigning node and edge potentials is the "standard" way to set up the BP algorithm. In Figure 18, the ROC curve labeled as "bp" shows the result of running standard BP on a day of traffic on November 1 from $ISP2$ (experiments on other traffic datasets produced similar results). As we can see from the figure, at an FP rate of 0.2%, Segugio, with all features, which

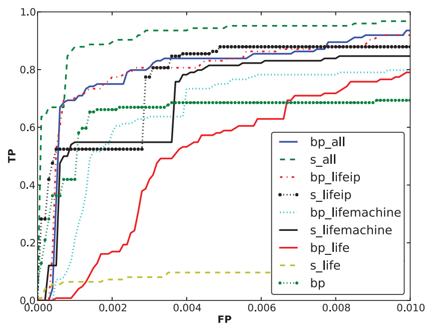


Fig. 18. Comparison between Segugio and Belief Propagation (FP rate is in [0, 0.01]).

is shown in the curve labeled "s_all," yields a TP rate close to 90%, while "bp" achieves a much lower TP rate of about 67%.

As outlined earlier, we can see that the standard loopy BP performs significantly worse than our Segugio system. To try to improve the results obtained via BP, we investigated different methods for assigning the node and edge potentials. For example, we learned the potentials for all nodes and edges directly from the data, using state-of-the-art methods (derived from Song et al. [2011]). Specifically, the potential of labeled nodes (both domains and machines) is set the same way as in the standard BP. However, to assign the node potential of unknown domain nodes, we use the output of Segugio. As discussed before, for each unknown domain d, Segugio outputs a score (a probability in $[0, 1]$) representing Segugio's confidence about the true nature of d. This score is used to set the initial node potential for d. The unknown machines' node potentials are assigned based on an estimation of the number of good and bad machines in the graph. For example, if 20% of labeled machines are known to be infected, then we assign a potential of 0.2 to all unknown machines. This indicates that we expect a machine to be bad with 0.2 probability.

The BP algorithm also requires each edge in the graph to have a set of edge weights (potentials) for either direction, that is, in the direction of machines to domains and domains to machines. These edge weights are also computed using the labeled data from the graph. For example, to compute the edge weights in the domain to machine direction, we proceed as follows. Considering all the edges where the two end nodes are labeled, we compute the following probabilities and use them as the edge weights: $P(m_i = good|d_j = good)$, $P(m_i = good|d_j = bad)$, $P(m_i = bad|d_j = good)$, and $P(m_i = bad|d_j = bad)$. As a result, the BP algorithm will use these four weights whenever it needs to propagate belief from a domain to a machine. The edge weights for the machine to domain direction are estimated the same way.

The comparison results are summarized in Figure 18. Each ROC curve shows a different configuration of our experiments. The "s_all" curve is obtained by using Segugio with all the features. This version of Segugio is then used to set the node potentials for the BP algorithm, which produces an ROC curve shown by "bp_all." As another example, the "s_lifeip" curve is a Segugio classifier trained by the domain activity (life) and IP-space features (without machine-behavior features). This classifier is used to assign node potentials for the unknown domains in the graph; then, the BP algorithm is run using these node potentials to produce the output shown by the "bp_lifeip" curve. By comparing all the different results, it is clear that Segugio outperforms the loopy BP algorithm in all possible configurations. One other great advantage of Segugio is its efficiency. While in our experiment, it took about 14 hours to run standard BP, Segugio required about an hour to train the behavioral classifier and produce classification for the test samples (see Section 5.8).

7. LIMITATIONS AND DISCUSSION

Segugio requires preliminary ground truth to label a set of "seed" known malware and benign nodes. Fortunately, some level of ground truth is often openly available, as in the case of public C&C blacklists and popular domain whitelists, or can be obtained for a fee from current security providers (in the case of commercial blacklists). In Section 5.6, we show that even using only ground truth collected from public sources, Segugio can still detect new malware-control domains. Note also that, while the ground truth may contain some level of noise, it is possible to apply some filtering steps to reduce its impact (e.g., see discussion in Section 3).

Because Segugio focuses on detecting "malware-only" domains, an attacker may attempt to evade Segugio by somehow operating a malware-control channel under a legitimate and popular domain name. For example, the malware owner may build a C&C channel within some social-network profile or by posting apparently legitimate blog comments on a popular blog site. While this is possible, popular sites are often patrolled for security flaws, which exposes the C&C channel to a potentially more prompt takedown. This is one of the reasons why attackers often prefer to point their C&C domains to servers within "bulletproof" hosting providers. Furthermore, Segugio is a DNS-based malware-control domain-detection system, which means it does not deal with pure P2P-based botnets as they do not use any centralized domains.

A possible limitation of Segugio is that a malware-control domain that is never queried by any of the previously known malware-infected machines is more difficult to detect. However, in Section 5.5, we showed that, by combining the *machine behavior* features (defined in Section 2.1.3) to the *domain activity* and *IP abuse* features, Segugio is still able to detect many such new domains.

Attackers might try to "poison" different feature groups of Segugio's classifier to degrade its performance. For example, malware on infected hosts could start resolving benign domains (poisoning machine-behavior features), and attackers could start contacting their own malware-control domains to poison domain-activity features or pointing them to benign IPs. However, we have already observed some of these behaviors in our real-world data, such as malware contacting benign domains for various reasons or attackers testing the launch of their new domains. Despite all this, Segugio's detection accuracy is still high.

Unlike systems that specialize in DGA-based botnets [Antonakakis et al. 2012], Segugio only monitors domain names that resolve to valid IP addresses; thus, it does not detect NXDOMAINS. These domains are a by-product of the process used by DGA-based malware to discover the currently active C&C domain. The vast majority of DGA-generated domains never resolve (they are never registered). Segugio's main goal is to track the occurrence of new malware-control domains that actively resolve. This includes DGA-generated domains that do resolve (i.e., those few DGA domains that are actually registered by the attacker) and can therefore point a malware-infected machine to an active malware-control server. In other words, even though Segugio does not monitor NXDOMAINS, it is capable of detecting DGA-generated domains that actively point to C&C servers.

Segugio needs to be retrained periodically to avoid detection accuracy degradation. Based on the evaluation results reported in Section 5.10, the detection accuracy of Segugio starts to deteriorate after about 15 days without retraining. Retraining could be performed quickly and does not impose significant overhead (see Section 5.8). Sections 5.2 and 5.3, however, show that Segugio maintains its high detection performance with gaps of at most 15 days between training and deployment, even if the trained model is used in different ISP networks.

Another possible challenge is represented by networks that have a high DHCP churn if source IP addresses are used as the machine identifiers. High DHCP churn may cause some inflation in the number of machines that query a given (potentially malware-related) domain. However, we should consider that Segugio can independently be deployed by each ISP. Therefore, for deployments similar to ours, the ISP's network administrators may be able to correlate the DHCP logs with the DNS traffic to obtain unique machine identifiers that can be used for building the machine-domain graphs.

Segugio's detection reports are generated after a given observation time window (one day, in our experiments). Therefore, malware operators may try to change their malware C&C domains more frequently than the observation window, so that if the discovered domains are deployed in a blacklist, they may be of less help for enumerating the infected machines in a network. However, it is worth noting that Segugio can detect both malware-control domains and the infected machines that query them at the same time. Therefore, infections can still be enumerated, thus allowing network administrators to track and remediate the compromised machines.

Furthermore, to concretely fight malware infections, the domains discovered by Segugio can be taken down (with the help of law enforcement), sinkholed, or blackholed. In this case, even though the machines remain infected, the malware installed on the machines will effectively be "disabled" as their C&C channel has been blocked. In our experiments, we also show that, in practice, the new malware-control domains detected by Segugio can be successfully added to domain blacklists to enable the detection of new infected machines several days or even weeks after the new domains were first discovered. Remedial actions can be performed on infected machines; however, in large ISP networks, it is highly likely that new malware-control domains will continuously emerge, due to new machines becoming infected every day. Segugio's goal is to help network security operators keep pace with new infections so that they can be promptly detected and mitigated.

Some ISP networks may host clients that run security tools that attempt to continuously "probe" a large list of malware-related domains, for example, to actively keep track of their activities (e.g., whether they are locally blacklisted, what is their list of resolved IPs, their name server names, and so on). Such clients may introduce noise into our bipartite machine-domain graph, potentially degrading Segugio's accuracy and performance. During our experiments, we used a set of heuristics to verify that our filtered graphs (obtained after pruning, as explained in Section 2) did not seem to contain such "anomalous" clients.

8. RELATED WORK

In Section 1, we discussed the main differences with recent previous work on detecting malicious domains, such as Antonakakis et al. [2010], Bilge et al. [2011], Antonakakis et al. [2011], and Manadhata et al. [2014]. In this section, we discuss the differences between Segugio and other related works.

Botnet/Malware detection: Pleiades [Antonakakis et al. 2012] is a recently proposed system that aims to detect machines infected with malware that makes use of domain generation algorithms (DGAs). While Pleiades monitors the DNS traffic between network users and their local DNS resolver, as we do, it focuses on monitoring nonexistent (NX) domains, which are a side-effect of DGA-based malware. Our work is different, because we do not focus on DGA-based malware. In fact, Segugio only monitors "active" domain names, and aims to detect malware-control domains in general, rather than being limited to detecting only DGA-generated domains.

Studies such as Gu et al. [2007, 2008a, 2008b], Yen and Reiter [2010] and Zhang et al. [2011] focus on detecting bot-compromised machines. For example, BotSniffer [Gu et al. 2008b] and BotMiner [Gu et al. 2008a] look for similar network behavior across network hosts. The intuition is that compromised hosts belonging to the same botnet share common C&C communication patterns. These systems typically require monitoring all network traffic (possibly at different granularities) and are therefore unlikely to scale well to very large ISP networks. Our work is different, because we focus on a more lightweight approach to detecting malware-control domains by monitoring DNS traffic in large ISP networks.

A large body of work has focused on detecting malware files. One method related to ours is Polonium [Chau et al. 2011], which aims to detect malware files using graphical models. Our work is different from Polonium in many respects. We focus on detecting new malware-control domains, rather than malware files. In addition, Polonium employs a very expensive loopy BP algorithm on a graph with no annotations. Furthermore, through pilot experiments using GraphLab [Low et al. 2010] we found that the inference approach used in Polonium would result in a significantly lower accuracy for Segugio with a huge negative impact on performance.

Malware C&C modeling and tracking. Wurzinger et al. [2009] propose to first detect malicious network activities (e.g., scanning and spamming) generated by malware executed in a controlled environment (see Egele et al. [2008]), then to analyze the network traffic "backwards" to find what communication could have carried the command that initiated the malicious activities. Jackstraws [Jacob et al. 2011] executes malware in an instrumented sandbox to generate "behavior graphs" for system calls related to network communications. These system-level behavior graphs are then compared to C&C graph templates to find new C&C communications. Our work is different, because we do not rely on performing detailed malware dynamic analysis in a controlled environment. Rather, we focus on detecting new malware-control domains via passive DNS traffic analysis in live ISP networks.

Sato et al. [2010] performed a preliminary study of unknown domains that frequently co-occur with DNS queries to known C&C domains. While the co-occurrence used in Sato et al. [2010] has some resemblance to Segugio's machine-behavior features, our work is different from theirs. For example, the system presented in Sato et al. [2010] suffers from a large number of false positives, even at a fairly low true-positive rate. Furthermore, unlike Segugio, Sato et al. [2010] are not able to detect new C&C domains that have low or no co-occurrence with known malicious domains. It is important to note that the work of Sato et al. [2010] has been evaluated only at a very small scale. In contrast, we performed a thorough evaluation of Segugio in many different settings, including cross-validation, cross-day and cross-network tests, feature analysis, performance evaluation, and direct comparison with Notos [Antonakakis et al. 2010] and BP algorithm. All of our experiments were conducted at large scale, via a deployment in multiple real-world ISP networks hosting millions of users.

Signature-based C&C detection. Researchers have recently proposed a number of studies that focus on a signature-based approach to detect malware C&C communications and the related malware C&C domains. For example, Perdisci et al. [2010] proposed a system for clustering malware that requests similar sets of URLs, and to extract token-subsequence signatures that may be used to detect infected hosts. ExecScent [Nelms et al. 2013] is a new signature-based C&C detection system that builds control protocol templates (CPT) of known C&C communications, which are later used to detect new C&C domains. Another recent signature generation system, called FIRMA [Rafique and Caballero 2013], can be used to detect C&C communications and the related malware-control domains.

These signature-based approaches typically require access to all TCP traffic crossing a network to enable the detection of C&C communications. Instead, our system is based on a much more lightweight monitoring of DNS traffic only.

Other related work. Karagiannis et al. [2005] consider *who is talking to whom* to discover communities among hosts for flow classification purposes. In a related study, Xu et al. [2011] use a bipartite graph of machine-to-machine communications. They use spectral clustering to identify groups of hosts with similar network behaviors. Coskun et al. [2010] use a graph-based approach to discover peer nodes in peer-to-peer botnets. While we also leverage bipartite graphs, our work is very different from Karagiannis et al. [2005], Xu et al. [2011] and Coskun et al. [2010] in both the goals and approach. Felegyhazi et al. [2010] take a proactive blacklisting approach to detect likely new malicious domains by leveraging domain registration information. Segugio is different since it mainly focuses on detecting new malware-control domains based on *who is querying what*. While we use information such as domain activity, Segugio does not rely on domain registration records.

9. CONCLUSION

In this article, we presented *Segugio*, a novel defense system that is able to efficiently discover new *malware-control* domain names by passively monitoring the DNS traffic of large ISP networks.

We deployed Segugio in three large ISP networks, and showed that Segugio can achieve a true-positive rate above 94% at less than 0.1% false positives. In addition, we showed that Segugio can detect control domains related to previously unseen malware families, and that it outperforms reputation systems based on BP algorithm like [Manadhata et al. 2014]. Furthermore, we demonstrated that Segugio can be used to detect previously unknown malware-infected machines in large ISP networks.

REFERENCES

Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. 2010. Building a dynamic reputation system for DNS. In *Proceedings of the 19th USENIX Conference on Security (USENIX Security'10)*.

Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nikolaos Vasiloglou, II, and David Dagon. 2011. Detecting malware domains at the upper DNS hierarchy. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*.

Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. 2012. From throw-away traffic to bots: Detecting the rise of DGA-based malware. In *Proceedings of the 21st USENIX Conference on Security Symposium (Security'12)*. USENIX Association, Berkeley, CA, 24–24. http://dl.acm.org/citation.cfm?id=2362793.2362817

Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. 2011. EXPOSURE: Finding malicious domains using passive DNS analysis. In *NDSS*. The Internet Society.

Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1, 5–32.

Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. 2011. Measuring pay-per-install: The commoditization of malware distribution. In *Proceedings of the 20th USENIX Conference on Security (SEC'11)*. USENIX Association, Berkeley, CA, USA, 13–13.

D. H. Chau, C. Nachenberg, J. Willhelm, A. Wright, and C. Faloutsos. 2011. Polonium: Tera-scale graph mining and inference for malware detection. *Proceedings of SIAM International Conference on Data Mining (SDM'11)* 131–142.

Baris Coskun, Sven Dietrich, and Nasir Memon. 2010. Friends of an enemy: Identifying local members of peer-to-peer botnets using mutual contacts. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 131–140.

Manuel Egele, Theodoor Scholte, Engin Kirda, and Christopher Kruegel. 2008. A survey on automated dynamic malware-analysis techniques and tools. *ACM Computing Surveys* 44, 2, Article 6.

Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. LIBLINEAR: A library for large linear classification. *The Journal of Machine Learning Research* 9, 1871–1874.

Mark Felegyhazi, Christian Kreibich, and Vern Paxson. 2010. On the potential of proactive domain blacklisting. In *Proceedings of the 3rd USENIX Workshop on Large-scale Exploits and Emergent Threats (LEET'10)*.

Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. 2008a. BotMiner: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *Proceedings of the 17th Conference on Security Symposium (SS'08)*. USENIX Association, Berkeley, CA, 139–154.

Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, and Wenke Lee. 2007. BotHunter: Detecting malware infection through IDS-driven dialog correlation. In *Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium (SS'07)*. USENIX Association, Berkeley, CA, Article 12.

Guofei Gu, Junjie Zhang, and Wenke Lee. 2008b. BotSniffer: Detecting botnet command and control channels in network traffic. In *Proceedings of the 15th Annual Network and Distributed System Security Symposium (NDSS'08)*.

Gregoire Jacob, Ralf Hund, Christopher Kruegel, and Thorsten Holz. 2011. JACKSTRAWS: Picking command and control connections from bot traffic. In *Proceedings of the 20th USENIX Conference on Security*. Berkeley, CA.

Thomas Karagiannis, Konstantina Papagiannaki, and Michalis Faloutsos. 2005. BLINC: Multilevel traffic classification in the dark. In *Proceedings of the 2005 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications (SIGCOMM'05)*. ACM, New York, NY, 12.

Daphne Koller and Nir Friedman. 2009. *Probabilistic Graphical Models: Principles and Techniques*. The MIT Press, Cambridge, MA.

Marc Kührer, Christian Rossow, and Thorsten Holz. 2014. Paint it black: Evaluating the effectiveness of malware blacklists. In *Research in Attacks, Intrusions and Defenses*. Springer, 1–21.

Ludmila I. Kuncheva. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. Wiley-Interscience, Hoboken, NJ.

Yucheng Low, Joseph Gonzalez, Aapo Kyrola, Danny Bickson, Carlos Guestrin, and Joseph M. Hellerstein. 2010. GraphLab: A new parallel framework for machine learning. In *Conference on Uncertainty in Artificial Intelligence (UAI)*. Catalina Island, CA.

Pratyusa K. Manadhata, Sandeep Yadav, Prasad Rao, and William Horne. 2014. Detecting malicious domains via graph inference. In *Computer Security - ESORICS'14*, Miroslaw Kutylowski and Jaideep Vaidya (Eds.). Lecture Notes in Computer Science, Vol. 8712. Springer, Berlin, 1–18.

Terry Nelms, Roberto Perdisci, and Mustaque Ahamad. 2013. ExecScent: Mining for new C&C domains in live networks with adaptive control protocol templates. In *Proceedings of the 22nd USENIX Conference on Security*. USENIX Association, 589–604.

Roberto Perdisci, Wenke Lee, and Nick Feamster. 2010. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*.

M. Zubair Rafique and Juan Caballero. 2013. FIRMA: Malware clustering and network signature generation with mixed network behaviors. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses*. St. Lucia.

Babak Rahbarinia, Roberto Perdisci, and Manos Antonakakis. 2015. Segugio: Efficient behavior-based tracking of malware-control domains in large ISP networks. In *Proceedings of the 2015 IEEE/IFIP International Conference on Dependable Systems &Networks (DSN'15)*.

Christian Rossow, Christian Dietrich, and Herbert Bos. 2013. Large-scale analysis of malware downloaders. In *Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 42–61.

Kazumichi Sato, Keisuke Ishibashi, Tsuyoshi Toyono, and Nobuhisa Miyake. 2010. Extending black domain name list by using co-occurrence relation between DNS queries. In *LEET*.

Le Song, Arthur Gretton, Danny Bickson, Yucheng Low, and Carlos Guestrin. 2011. Kernel belief propagation. In *Artificial Intelligence and Statistics (AISTATS)*.

Symantec. 2013a. India Sees 280 Percent Increase in Bot Infections. Retrieved July 18, 2016 from http://www.symantec.com/en/in/about/news/release/article.jsp?pr id=20130428_01.

Symantec. 2013b. Internet Security Threat Report, Volume 18. http://www.symantec.com/content/en/us/enterprise/other_resources/b-istr_main_report_v18_2012_21291018.en-us.pdf.

Peter Wurzinger, Leyla Bilge, Thorsten Holz, Jan Goebel, Christopher Kruegel, and Engin Kirda. 2009. Automatically generating models for botnet detection. In *Proceedings of the 14th European Conference on Research in Computer Security (ESORICS'09)*.

Kuai Xu, Feng Wang, and Lin Gu. 2011. Network-aware behavior clustering of Internet end hosts. In *Proceedings of IEEE INFOCOM*.

Ting-Fang Yen and Michael K. Reiter. 2010. Are your hosts trading or plotting? Telling P2P file-sharing and bots apart. In *Proceedings of the IEEE 30th International Conference on Distributed Computing Systems (ICDCS'10)*.

Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. 2011. Detecting stealthy P2P botnets using statistical traffic fingerprints. In *Proceedings of the IEEE/IFIP 41st International Conference on Dependable Systems &Networks (DSN'11)*.

Received September 2015; revised June 2016; accepted June 2016