Measuring and Detecting Malware Downloads in Live Network Traffic

Phani Vadrevu[1], Babak Rahbarinia[1], Roberto Perdisci[1,2], Kang Li[1], and Manos Antonakakis[3]

[1] Dept. of Computer Science, University of Georgia, Athens, GA, USA
[2] School of Computer Science, Georgia Institute of Technology, Atlanta, GA, USA
[3] Damballa, Inc.
{vadrevu,babak,perdisci,kangli}@cs.uga.edu, manos@damballa.com

Abstract. In this paper, we present AMICO, a novel system for measuring and detecting malware downloads in live web traffic. AMICO learns to distinguish between malware and benign file downloads from the *download behavior* of the network users themselves. Given a labeled dataset of past benign and malware file downloads, AMICO learns a *provenance classifier* that can accurately detect future malware downloads based on information about where the downloads originated from. The main intuition is that to avoid current countermeasures, malware campaigns need to use an "agile" distribution infrastructure, e.g., frequently changing the domains and/or IPs of the malware download servers. We engineer a number of statistical features that aim to capture these fundamental characteristics of malware distribution campaigns.

We have deployed AMICO at the edge of a large academic network for almost nine months, where we continuously witness hundreds of new malware downloads per week, including many *zero-days*. We show that AMICO is able to accurately detect malware downloads with up to 90% true positives at a false positives rate of 0.1% and can detect zero-day malware downloads, thus providing an effective way to complement current malware detection tools.

1 Introduction

Drive-by downloads and social engineering attacks have become one of the most prevalent ways through which machines are compromised with malicious software, or *malware* [10, 17, 19]. As a consequence, by simply browsing the Web, users (or their browsers) may be either forced or lured to download and run malware samples, effectively relinquishing control of their machines to the attackers.

Users often rely on host-based anti-virus software (AVs) to protect themselves from malware infections. However, it is known that AVs are only partially effective due to the sophisticated code polymorphism techniques adopted by malware authors, and are not capable of protecting users from the latest threats [13]. To compensate for this detection gap, modern browsers make use of URL blacklists, such as Google Safe Browsing [8] (GSB). Essentially, GSB maintains a large list of domain names and URLs that are known to be related to malware downloads. Therefore, every time the user visits a URL, before the browser fetches the URL content, the GSB API is queried. If the URL is blacklisted, the browser stops loading the URL's content and the user will be notified, thus preventing a possible malware download. Unfortunately, by nature, static blacklists such as GSB also lag behind the threat, and suffer from a non-negligible number of false negatives, as we show in Section 4.6.

In this paper, we present AMICO[1], a novel system for measuring and detecting malware downloads in live web traffic using *download provenance* information (see Figure 1). Every time a network user downloads an executable file (we limit ourselves to Windows executables, in the current implementation), AMICO performs an on-the-fly reconstruction of the download from the network traffic, and copies the file to a download history database. In addition, the database stores information regarding *who* (i.e., what and how many machines) downloaded the file and *where* the download came from. By leveraging the (partial) ground truth provided by existing AV tools, we can label some of these downloads as either malware or benign. Using these labeled download events collected during an initial training period, AMICO learns the *provenance characteristics* of past malware and benign executable files from the *download behavior* of the network users themselves. This allows us to build a statistical classifier that, given a new file download and its related provenance information, is able to accurately classify whether the downloaded file is likely to be malicious or not. Unlike traditional AV products, AMICO does not rely on searching for signs of malicious code in the content of the downloaded files. Furthermore, the classification is performed independently of whether third-party detection results may exist about the new downloads, and can therefore be used to *complement existing malware defense* techniques (see Section 3 for details).

The intuitions that motivate us to leverage provenance information for detecting malware downloads are as follows. To avoid signature-based AV detection, malware authors make heavy use of code polymorphism. Therefore, victim machines infected with the same malware may in fact have downloaded different "variants" of the same malware file. Consequently, a given malware file may be downloaded by only few machines. On the other hand, benign executable files are fairly "stable", and change only when a new release version is available. Therefore, benign files may be downloaded, in time, by several different clients.

Furthermore, to avoid static blacklists, malware distribution sites need to frequently relocate. For example, the attacker may register a large set of domain names that point to the distribution site. This allows for "advertising" the malware downloads (e.g., though email spam, drive-by download exploit servers, etc.) from frequently changing domains. Similarly, the IP address of the malware distribution server may periodically change (although more slowly, compared to the domain changes). On the other hand, benign executable files are typically hosted at professionally-operated service providers with a fairly stable domain

[1] Accurate Malware Identification via Classification of live network traffic Observations.

name and network infrastructure. Even when the benign files are distributed via content delivery networks (CDNs), both the domain name (especially the second-level domain) and the IP address or BGP prefix of the distribution server may be fairly stable, especially with respect to download requests originating from the same local network. This causes malware downloads to have a download source "footprint" that is noticeably different from benign downloads.

Once deployed, for each new executable file download event AMICO measures a number of provenance features specifically engineering to capture the above observations, and is able to accurately classify the downloads into benign or malicious. Notice also that while our current implementation of AMICO is designed to monitor the traffic from the edge of a network, nothing prevents us from deploying AMICO "within" a web proxy (e.g., using the ICAP protocol (RFC 3507)). This may be particularly useful in enterprise network environments, which typically already deploy a web proxy, and often perform SSL man-in-the-middle[1] to enable fine-grained inspection of encrypted traffic. This would allow AMICO to also observe possible file downloads over HTTPS, further increasing its coverage.

In summary, we make the following contributions:

– We present AMICO, a novel system that aims to efficiently measure and detect malware downloads in live network traffic. In contrast to static blacklists, AMICO builds a provenance classifier that can dynamically and accurately detect malware samples based on the download behavior of the network users.
– We have deployed AMICO at the edge of a large academic network serving tens of thousands of users for almost nine months. Our measurements show that, in spite of the widespread use of malware URL blacklists in modern browsers, we continuously witness hundreds of new malware downloads per week, including many zero-days. Surprisingly, a non-negligible number of malware downloads originate from even the most popular websites.
– We perform an extensive evaluation of AMICO's malware detection capabilities. The experimental results show that our provenance classifier is able to accurately detect malware downloads with up to 90% true positives at a false positives rate of 0.1%.

2 Related Work

Malware Detection: Oberheide et al. [13] highlight the limitations of signature-based AV tools, and propose a new system called CloudAV that leverages a combination of AV tools to improve malware detection coverage. Some researchers have proposed to improve the detection of malware file content using statistical machine learning techniques [9, 14, 15], rather than signature matching. Others have focused on measuring specific types of malware distribution tactics, such as rogue AV campaigns and pay-per-install (PPI) operations, or on measuring and

[1] For example, `http://crypto.stanford.edu/ssl-mitm/`

detecting drive-by malware downloads [5–7, 10, 16, 17, 21]. Our work is different, because we do not focus on the file content or drive-by downloads. Rather, AMICO aims to detect malware downloads in general by inspecting network traffic in real-time, and by leveraging download provenance information.

Domain Reputation: A number of systems that aim to detect malicious, low-reputation domain names have been proposed [2, 3]. These systems are able to detect malicious domains in general (e.g., spam domains, phishing sites, malware download sites, etc.), with particular emphasis on malware command-and-control (C&C) domains. Our work is different, because we specifically aim to detect malware file downloads. We correlate many different features that go beyond domain names and the IP addresses they resolve to, such as the file download features, URL features, and download request features. Furthermore, in Section 4.3 we show that domain reputation systems by themselves are not sufficient to accurately detect malware downloads.

Google CAMP: CAMP [18] detects malware domains based on a reputation score computed over a number properties of the download source (e.g., the domain name of the download server, the server IP, etc.). Although, AMICO and CAMP share similar goals, our AMICO system differs in many important aspects from CAMP. First of all, AMICO is *browser agnostic*, whereas CAMP is built within Google Chrome, and can only monitor downloads from Chrome users[1]. More importantly, CAMP is a *closed-source service*: all download information and decision rules are "owned" by Google, and a network administrator has no easy way to gain a complete picture about executable file downloads happening in his/her network. On the other hand, AMICO was designed to exactly fulfill this network admins' need, by offering *network-wide information about what clients in the monitored traffic are downloading malware files and from where.* This enables the administrators to promptly respond to security incidents and limit potential damage to other network assets. Furthermore, unlike in CAMP, by deploying AMICO the information about what machines may be infected will not leave the local network. This may be particularly important in highly sensitive enterprise or government networks, where shipping information such as visited URLs, downloaded files, and potential malware infections to a third-party may pose risks to the reputation of the institutions that operates the network.

AMICO and CAMP also differ with respect to their technical approach. For example, we measure several statistical features that are not used in CAMP, and employ a different, machine-learning-based approach.

3 System Description

In this section, we discuss the internals of our system. AMICO consists of three main components, shown in Figure 1: (1) the *download reconstruction* module, (2) the *download history* database, and (3) the *provenance classifier*. In the following, we provide details on how these components work.

[1] It appears that Microsoft may also have built a similar proprietary system specific to IE9 [12], although we were not able to find its technical details.

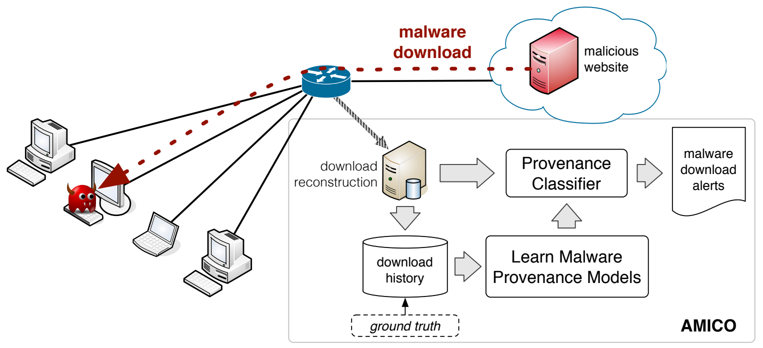


Fig. 1. AMICO System Overview

3.1 Reconstruction of Executable Files

The *download reconstruction* module aims to inspect all web traffic, and extract a copy of Microsoft Windows executable files that are being downloaded by the network users. To this end, AMICO monitors all traffic at the edge of a network, and performs efficient real-time TCP flow reconstruction using a custom-built multi-threaded software component. As TCP flows are being reconstructed, a traffic identification module keeps track of all HTTP flows, and discards the remaining non-HTTP traffic. HTTP request-response pairs are reconstructed on-the-fly, and the responses are inspected to determine whether they carry a portable executable (PE) file [11]. Every time a PE file is detected, AMICO copies the reconstructed response on persistent memory, along with the related HTTP request and some additional information, such as source and destination IPs and ports, and a timestamp. Sensitive information, such as source IP addresses, cookies, and certain HTTP headers, are either anonymized or removed outright, in accordance with policies set forth by our Institutional Review Board.

3.2 Download History Database

The *download history* database stores all information gathered by the download reconstruction module. In our current implementation, as soon as a downloaded file is stored, AMICO computes the SHA1 hash of the file and automatically queries VirusTotal (VT) [1], to determine whether the file had ever been scanned before and was found to be malicious by any AV. This is done merely for convenience, to avoid acquiring and running multiple local AV scanners.

It is important to notice that the information obtained from the AVs is necessary to build the ground truth used to label *past* download events and train the provenance classifier, as discussed in Section 3.3. However, to this end AMICO only submits the hash of downloaded files to VT, and does not need to submit the URL and `Referer` of the download events, which may be considered as more sensitive by the network administrator.

Notice also that if submitting the file hashes to a third-party services such as VT still represents a concern, the network administrator can "conceal" the origin of the file hashes by submitting them through a proxy located in a separate network. In alternative, submitting the file hashes can be avoided completely by scanning the downloaded files locally, using multiple different AV products. In this latter configuration, AMICO would prevent any leakage of information from the monitored network to third-parties.

3.3 Provenance Classifier

The provenance classifier aims to complement AV-based malware detection, by identifying malicious file downloads based on *how the file was downloaded*, rather than how the file "looks". To this end, we extract a number of provenance features that aim to capture the following facts: Has any of the network users ever downloaded the same file in the past? Has any executable file been downloaded from this domain name, server IP address, BGP prefix, etc.? If so, were the previously downloaded files malicious (or at least suspicious)?

We first give a description of the detection features used by AMICO, and then describe how the provenance classifier can be trained and deployed.

Provenance Features Let e be an executable file download event occurred at time t_e. Also, let F_e be the downloaded file, $Host_e$ be the domain name associated with the HTTP request for the file, URL_e be the URL of the request (i.e., the file path, file name, query string, etc.), and $ServIP_e$ be the IP address of the server from which the file was downloaded. We translate each such event into a *feature vector* $\vec{v}_e$ as follows. We first consider only *past* download events, namely events occurred at any time $t < t_e$, and measure the following main groups of features (a complete list of features is given in Appendix):

- **Past file downloads**: We measure four different features as follows: the number of times that the file F_e was downloaded in the past (we use the file's SHA1 to compute this more efficiently); the (estimated) number of distinct clients that downloaded that file; how many days ago was F_e downloaded for the first time; and how many times per day (in average) the client machines in the monitored network downloaded the same file F_e.
  Intuition: Many benign executable files are downloaded, in time, by several different clients. Also, their hash is typically very "stable" and only changes after a new version release. On the other hand, due to heavy polymorphism applied by malware developers to evade signature-based AV detection, the hash of a given malware will change frequently. Consequently, the same malware file will typically be downloaded by only few victims.
- **Domain features**: Let d_e be the domain name related to the download request, and let $2LD(d_e)$ be its effective second-level domain[1]. Overall, we

[1] For example, 2LD(www.bbc.co.uk) = bbc.co.uk. To compute the effective 2LDs we use the Mozilla public suffix list (`publicsuffix.org`) augmented with a large list of second-level domains related to dynamic-DNS providers.

measure a set of twenty-four features, twelve of which are related to past download events from d_e, and another twelve related to past downloads from any domain under $2LD(d_e)$ (i.e., any domain that matches $*.2LD(d_e)$). For example, we measure how many confirmed malware samples had been previously downloaded from d_e; the number of confirmed benign files from the same domain; the ratio between malware and benign downloads; the total number of executable downloads from d_e (including the "unknown" files that cannot be labeled either way), the average number of AV labels for the confirmed malware samples (i.e., how many different AVs flagged the file as malware), etc. We measure similar features for $2LD(d_e)$.
Intuition: To avoid static blacklists, attackers often register many different domain names that can be used to "advertise" the malware downloads. Each malware download domain is typically used for a short amount of time before it is replaced with a new one, and may therefore serve only a small number of malware downloads to a few victims. On the other hand, benign executable files are typically hosted at professionally-run service providers, and their server's domain names (or their second-level domains) are usually very stable, serving the same benign files to potentially many clients. Our domain features attempt to capture such intuitions.

– **Server IP features**: In a way similar to the *domain features*, we measure twenty-four different features, twelve of which are related to the $ServIP_e$ and another twelve to its BGP prefix, $BGP(ServIP_e)$. For example, we measure how many confirmed malware samples had been previously downloaded from $ServIP_e$; the number of confirmed benign files from the same IP; the ratio between malware and benign downloads; etc. We repeat the same measurements for $BGP(ServIP_e)$.
Intuition: While malware samples are heavily polymorphic, the network infrastructure used to distribute different variants of the same malware is usually somewhat more stable. This is particularly true for the server IP from which the downloads originated. In fact, while the attackers have a good level of flexibility regarding registering new domain names to be used for malware distribution, it is more difficult to change IP addresses with high frequency. Therefore, we may see more than one malware download from the same server IP, or the same BGP prefix.
– **URL features**: Given the URL_e related to the download, we only consider its path, file, and query string (i.e., we don't consider the domain name as being part of the URL). From URL_e we measure six different features. For example, we measure the number of total past file downloads that share the same URL, the number of confirmed distinct files downloaded from that URL, and the number of confirmed malware samples. Because URLs may change frequently, especially if they contain name-value pairs in the query string, we also measure similar features related to the *URL structure*. For example, one way to derive the URL structure is to replace all alphanumeric characters with wildcards, keeping special characters such as '/', '.', '?', '=', '&', ':', ';',etc. We can then measure the total number of past downloads that share the same URL structure, the number of confirmed malware, etc.

Intuition: The intuition here is that, unlike for benign downloads, malware URLs may change frequently to avoid blacklists. Furthermore, we noticed several malware distribution campaigns that advertise many different download URLs with a similar "anomalous" structure, compared to URLs used in benign file downloads. Therefore, if the current download's URL_e has the same structure as URLs used in several past malware downloads, we should increase the likelihood that URL_e is also related to a malware download. Our URL features attempt to capture these observations.

– **Download request features**: In addition to the features described above, which *look into the past*, we measure five different features that *look at the present* single download event e. We check whether the header of the HTTP request that initiated the download contained a valid domain name in the `Host` field, and whether a `Referer` URL was or not present. Also, we consider the file extension (if any) extracted from the URL_e as a feature, and we measure the total length of the URL and the "depth" for the URL path (e.g., `/a/b/c/d.exe` has a depth of four).
Intuition: These features are justified by the fact that we empirically observed many cases in which malware download requests do not carry any `Referer` string, or may report an IP address in the `Host` field, instead of a domain. Also, the URLs for malware downloads often "look" visually different from the URLs related to benign downloads (e.g., the URL may have a `.gif` or `.jpg` extension, although it serves an executable file). The download request features attempt to capture these observations.

It is worth noting that none of the features (or groups of features) described above are sufficient by themselves to accurately distinguish between malware-related and benign file downloads. However, as we show in Section 4, each group provides a meaningful contribution. Furthermore, in combination they yield high classification accuracy. In Section 4, we also show that the overall accuracy does not heavily depend on one single group of features. In turn, this makes AMICO more robust to evasion, because an attacker would need to evade different types of features at the same time, as discussed more in details in Section 5. In addition, evading some groups of features such as the server IP and domain features, may require the attacker to make heavy changes to her malware distribution infrastructure, thus causing the attacker to incur a significant cost.

Training and Deploying the Classifier To build the provenance classifier, so that AMICO can automatically distinguish between benign and malware downloads, we take a supervised learning approach. That is, we first collect a set of *labeled* executable file download events, and use this initial *training set* to learn the malware provenance models, as shown in Figure 1. The process used to label the download events is described in Section 4.2.

The training phase proceeds in two high-level steps. We first monitor the network traffic for a period of time T_f (one month, in our experiments), and record a "bootstrap" set of download events. Essentially, the information collected during T_f allows us to measure our detection features over new download events (i.e., events that occur after T_f). After this initial T_f period, we are ready to collect new downloads and compute the feature vectors necessary for training the provenance classifier. To this end, we collect new download events for an additional period of time T_{train} (two months, in our experiments). For each new download event e at time t_e, we compute the related features using information about all downloads observed until $t < t_e$. We label the download events gathered during T_{train} for which ground truth is available (see Section 3.2), and use their feature vectors to train the *provenance classifier*.

Once the classifier is trained, any new executable file downloaded after $(T_f + T_{train})$ can be translated into a feature vector and classified by AMICO into either *malware* or *benign* using the provenance classifier. Specifically, for each new download event e, the classifier will output a *malware score*, $s(e)$. If the score $s(e)$ is greater than a given threshold, AMICO labels the downloaded file as *malware*, and raises an alert (e.g., notifies the network administrator). The detection threshold can be chosen during training to produce the desired trade-off between true and false positives. In Section 4 we show that AMICO can achieve high true positives even at very low false positive rates.

It is important to notice that for each download event e observed at time t_e, we extract its features by *looking back at past downloads and their related past ground truth*. Namely, we only consider information about downloads observed at any time $t < t_e$. To classify the new event e, we rely solely on the output of the provenance classifier, and do not use any information from external sources about e itself. Specifically, we do not consider any information that may be obtained from VirusTotal or GSB about e. In Section 4.6, we show that AMICO can in fact complement popular AV- or blacklist-based malware defense tools, because it can detect many malware downloads missed by third-party systems.

4 Evaluation

4.1 Implementation and Deployment

We implemented AMICO using different languages. We custom-built the download reconstruction module in C, to achieve high efficiency. Preliminary performance experiments show that one single instance of the reconstruction module can sustain over 300Mbps of traffic on commodity hardware. However, a machine with a multi-core CPU can run multiple instances independently on different network interfaces. For example, in our experiments, we run two instances of the reconstruction module that receive traffic from two different traffic mirroring sources on the same machine. We are currently monitoring over 600Mbps of traffic (during peak time) from the campus-wide WiFi network of our entire academic institution. The datasets and experiments we discuss below are derived from approximately nine months of traffic, from July 2012 to March 2013.

The components of AMICO used to store the download events, extract the detection features, and collect the ground truth, are written in Python. To build the provenance classifier we use the *Random Forest* algorithm [4] implemented in Weka [20], because it can be trained very efficiently and performs competitively compared to more complex algorithms. Our prototype implementation of AMICO is available under open-source license at `http://www.cs.uga.edu/~perdisci/amico/`.

4.2 Experimental Setup

Measuring the Detection Features To measure the detection features used by AMICO, we follow the definitions given in Section 3.3. It is worth remembering that some features measure things such as the number of malware, benign, and total samples downloaded *in the past* from a given domain name or server IP address, for example. To label past download events we proceed as follows. During deployment, for each file download captured by AMICO, we compute its SHA1 hash, and immediately submit the hash to VirusTotal (VT). If VT returns "unknown" as an answer, i.e., nobody has ever submitted a file with that hash before, we mark the downloaded file as *unknown to VT*. It is worth remembering here that, as we discussed in Section 3.2, if submitting the file hashes to a third-party system such as VT was a concern, we could simply scan the downloaded files *locally* using multiple different AVs. In our current prototype, we chose to rely on VT mainly because it made our system deployment easier.

Every file analyzed by VT is scanned with more than 40 different AV products. However, we noticed that some of the AV scanners produced a non-negligible number of false positives (e.g., marking some well known benign executable files as malware). We noticed this was especially true for less well-known AV products. Therefore, we decided to consider a "trusted" subset of nine AV products[1] that are very well known, and cover a large AV market share. Given this set of trusted AVs, we use the following labeling rules on VT's results:

Labeling Rules
1) if the SHA1 of the file was not present in VT, label the file as *unknown to VT*, otherwise
2) label the file as *malware* if two or more "trusted" AVs flagged the file as malware;
3) label the file as *benign* if none of the AVs (either trusted or non-trusted) flagged the file as malware;
4) label the file as *suspicious* in all other cases.

Notice that when measuring the features from a new download event e observed at time t_e, we only use the labels obtained from VT's output over *past* download events, i.e., at any time $t < t_e$. In other words, we do not use any third-party information (from AVs or blacklists) related to e itself.

Establishing the Ground Truth To evaluate the results of our deployment of AMICO, we need to collect *clean*, reliable ground truth that contains as little

[1] Avast, AVG, McAfee, F-Secure, Kaspersky, Sophos, Microsoft, TrendMicro, and Symantec.

noise as possible. To achieve this goal, we proceed as follows. If a download was labeled as *unknown to VT*, we submit the file ourselves to VT (we only submit samples that pass a number of privacy-preservation criteria). For each sample that was either already present in VT or submitted by us, after one month time we send a "re-scan" request to VT, so that the same file is scanned again by all AVs. The intuition is that it may take some time for AV companies to build signatures for new malware samples [13]. Therefore, even though a malware may be missed by the AVs at the time of submission, after one month it is likely that the AV companies may have developed the necessary detection signatures (we did observe several of such AV label changes during our study).

It is important to notice that the one-month VT re-scan procedure described earlier is used only for the purpose of collecting the ground truth "externally" to AMICO, to enable a more reliable evaluation of our system's accuracy. The re-scan information is not used for the purpose of measuring the statistical features used by the provenance classifier.

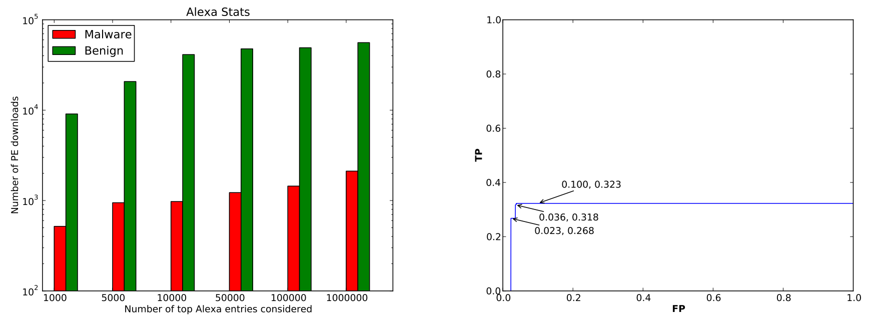
Cross-Validation Dataset (CVD) To collect the dataset of labeled downloads for cross-validation (see Section 4.4), we proceeded as follows. We first collected one month (T_f) of "bootstrap" download events, to enable the measurement of the detection features for future downloads. We then collected new download events and computed the related feature vectors for the remaining eight months. To label these feature vectors, we used the ground truth obtained as explained above. Overall, we obtained 55,396 *benign*, and 4,928 *malware* feature vectors (the *suspicious* samples are excluded from the cross-validation).

Training Dataset (TRD) Besides cross-validation tests, we also performed an evaluation of a real-world deployment of AMICO (see Section 4.5). To this end, we followed the guidelines discussed in Section 3.3. Like for **CVD**, we first collected one month (T_f) of initial download events. Then, we further collected new download events and the related feature vectors for the following two months (T_{train}), and we used these two-month data as a training dataset for the provenance classifier. To label the feature vectors in the training dataset, we used the ground truth gathered as explained earlier. Overall, the training data contained 16,074 *benign*, and 1,273 *malware* feature vectors.

Test Dataset (TSD) The test dataset consists of all download events collected after the first three months ($T_f + T_{train}$) necessary to gather the "bootstrap" download events and the training data. Essentially, this dataset consists of the last six months of download observations. Overall, the test dataset contained 39,322 *benign*, and 3,655 *malware* feature vectors.

4.3 Live Traffic Measurements

In this section we report a number of measurements on the traffic observed during our deployment of AMICO. Notice that here we discuss findings related



(a) (b)

Fig. 2. Downloads from top Alexa domains (a); Detection results using Notos (b).

only to the *download reconstruction* module and *download history* database. We defer the evaluation of the provenance classifier (and the entire AMICO system) to Sections 4.4 and 4.5.

During deployment, our sensors observed web traffic from several thousands distinct source IP addresses per day within our network. Table 1 reports a number of statistics about the reconstructed download events. To label the downloads we applied the "clean" ground truth labeling rules discussed in Section 4.2. Overall, we observed an average of 24 confirmed malware downloads per day, 67 daily suspicious downloads, and 253 daily benign downloads.

	Malware		*Suspicious*		*Benign*	
	Total	*Daily Avg.*	*Total*	*Daily Avg.*	*Total*	*Daily Avg*
Download events	5,326	24	15,665	67	59,988	253
Distinct files	1,893	12	2,879	38	5,849	112
Distinct domains	849	10	1,009	27	1,338	43
Distinct server IPs	1,009	6	2,186	41	2,776	59

Table 1. Overall live network download statistics for executable files

Figure 2(a) reports the number of confirmed malware and benign downloads that we observed from the top 1,000, 5,000, 10,000, 100,000, and 1M most popular domains[1], according to Alexa (`alexa.com`). Surprisingly, we found that about 18% of all confirmed malware downloads originated from the top 10,000 domains, and we also observed 518 malware downloads originating from the top 1,000 domains. After investigating, we noticed that these malware samples were downloaded from websites related to software distribution, file sharing, and cloud services (e.g. `softonic.com`, `hotfile.com`, `amazonaws.com`, `cloudfront.net`, etc.). Furthermore, we found that 40% of the domains from which benign downloads originated were "unpopular" sites outside of the top 1M rank. These two

[1] The list of domains we consider reports second-level domains.

facts make it difficult to implement a purely whitelist-based approach to preventing malware downloads, because such an approach would likely cause a non-negligible number of false positives and false negatives.

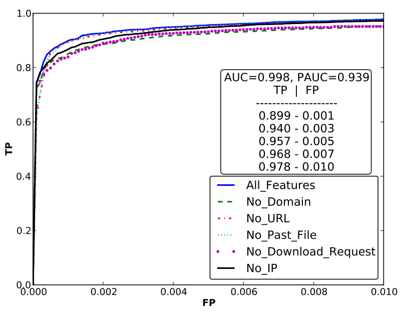
The above results may also have an impact on domain reputation systems. Therefore, we performed experiments to verify if dynamic domain reputation systems, in particular Notos [2], would be sufficient to block malware downloads. To this end, we fed all (domain name, server IP) pairs related to the malware downloads observed by AMICO. We then queried Notos to obtain a reputation score for each such pair. By varying a detection threshold over Notos' scores, we obtained the ROC curve in Figure 2(b) (notice that the FP rate is in [0,1]). As we can see, Notos has a relatively low detection rate for malware download domains. The reason why the ROC in Figure 2(b) is flat, is because many samples were "rejected" by Notos and assigned a score of zero, because Notos did not have enough information to compute a reliable score. We believe this is due to the fact that the reputation system is "biased" towards accurately detecting malware command-and-control (C&C) domains, rather than malware download servers. It is also worth noting that the version of Notos we used was trained on domain reputation information from a different network environment, and this may be another cause for the relatively low detection rate.

From the above discussion we can extrapolate two important observations: (1) using only domain name information to detect malware downloads may not be sufficient; and (2) training on the traffic specific to the deployment environment may yield better detection results. This further motivates the approach taken by AMICO, which *learns many different types of features related to the download behavior of the users in the monitored live network.*

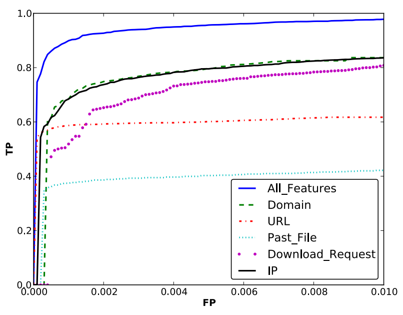
4.4 Cross-Validation

To evaluate AMICO's malware detection accuracy, we perform 10-fold cross-validation tests using the entire **CVD** data (see Section 4.2). Figure 3(a) shows the ROC curve we obtained by using all features, as well as the ROC curves obtained by removing one of the feature groups described in Section 3.3 at a time. Note that we only plot the *partial* ROC for false positive rates ranging from 0 to 1%, to highlight the classifier performance at low false positives. The small table embedded in Figure 3(a) provides the trade-off between the true positive (TP) rate and false positive (FP) rate for some selected operation points on the "all features" ROC. It also reports the normalized area under the ROC and partial ROC curves (AUC and PAUC). As we can see, we can achieve a TP rate close to 98% at 1% FP rate. Furthermore, when we tune the detection threshold to achieve a FP rate of 0.1%, the classifier still yields an TP rate close to 90%, using all features. In addition, from Figure 3(a) we can see that the provenance classifier is not overly reliant on a single group of features.

Figure 3(b) shows the results obtained using only one group of features at the time. As we can see, each feature group gives a meaningful contribution to accurately detecting malware downloads, with the server IP and domain features providing the largest single contributions.



(a) leaving one feature group out



(b) one feature group only

Fig. 3. 10-fold cross-validation results (FPs in [0,1%])

4.5 Train-Test Experiments

In this section, we discuss experiments performed to demonstrate the accuracy of AMICO in a real-world deployment setting. To this end we train the provenance classifier over the **TRD** dataset, and test it on the remaining **TSD** dataset. Figure 4 reports the ROC curve computed by using a provenance classifier built with all available features, as well as ROC curves related to separate tests in which we eliminate one group of features at a time. Like for the cross-validation results, we see that AMICO can achieve more than 90% TP rate for an FP rate $\leq$ 1%. Also, we can see that no one particular feature group is critical to obtaining good classification results.

Table 2 highlights the classification results for new malware downloads characterized by a never-before-seen file (*unseen SHA1*), and/or domain (*unseen domain*), and/or server IP (*unseen sever IP*). The "correct" column reports the number of correctly classified malware downloads. The detection threshold is set so to keep the overall FP rate (measured on benign **TSD** downloads) below 1%. As we can see, even when AMICO observes a completely new file from a new source (domain or server IP), it can still accurately classify the download event.

4.6 New Findings

In this section we discuss how AMICO can successfully complement existing malware detection approaches, such as AV scanners and static URL blacklists. All results discussed below consider a configuration of AMICO's detection threshold that yields an FP rate $\leq$ 1%.

1) Malware "unknown" to VT: Of the 3,655 confirmed *malware* downloads in the test dataset **TSD** (see Section 4.2), 1,031 malware samples were initially *unknown to VT*. That is, the first time we submitted the file's SHA1 to VT, the file was not present in VT's database. Of these, AMICO correctly classified 974 (94.5%) as malware at the time of download.

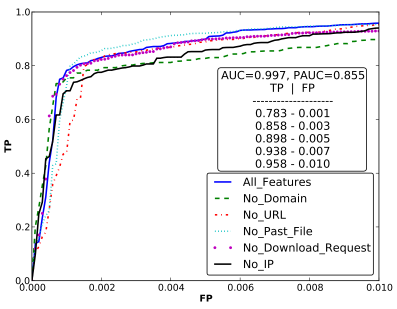


Fig. 4. Performance of Provenance Classifier on all test instances (FPs in [0,1%])

Unseen Feature	*Correct %*
SHA1	90% (895/994)
Domain	85% (360/422)
Server IP	93% (1139/1222)
SHA1 & Domain	85% (328/386)
SHA1 & Domain & Server IP	85% (295/346)

Table 2. Detection of "unseen" malware downloads (FP=1%)

2) Zero-day malware: We also found 187 malware downloads for which all nine "trusted" AV scanners in VT initially classified the file as benign (i.e., no AV label was attributed to the files, the first time they were scanned), and then were labeled as malware after the one-month re-scan (see Section 4.2). Therefore, we regard these file downloads as *zero-days*. Of these 187 zero-day malware downloads, AMICO classified correctly 147 (78.6%).
3) Static blacklists: For each download event observed by AMICO, we queried GSB at the time of the download to see if the domain name or URL associated to the download was present in the blacklist (we query GSB only for the purpose of enabling a comparison between our system and URL blacklists). Surprisingly, we found that out of the 3,655 malicious downloads, at the time of download GSB failed to detect 3,562 (97.5%). We believe this apparently high false negative rate is likely due to the fact that many potential malware downloads are already blocked by GSB, and therefore cannot be observed in the network traffic. However, many malware downloads that evade GSB's static blacklist are observable in the traffic, and can be captured by AMICO. Of the 3,562 malware downloads missed by GSB, AMICO correctly detected 3,412 (95.8%).

5 Limitations

Our current implementation of AMICO focuses on inspecting HTTP traffic, because we mainly target malware downloads that happen via the browser. To evade AMICO, malware developers may attempt to propagate their malware samples over HTTPS, thus "hiding" the executable files from AMICO's reconstruction module. However, it is worth noting that switching to HTTPS may have some drawbacks for the attacker. For example, because the domain names associated with the malware distribution servers have to change frequently, to avoid static blacklists, for each new domain the attacker would have to purchase a signed SSL certificate from a certificate authority (CA), thus incurring a non-negligible cost. In alternative, the attacker may use self-signed certificates. However, in this case the browser will typically alert the user of a potential security problem, thus possibly scaring away a large fraction of potential victims.

Furthermore, AMICO could be deployed "within" a web proxy that performs SSL man-in-the-middle.

It may be possible for sophisticated exploit code to force the browser to download an encrypted PE file, which can then be decrypted before the original malware file is executed. This scenario is analogous to detecting malware updates initiated from an infected machine, in which the method for downloading the files can be (almost) freely chosen by the already running malware instance. We therefore consider this scenario outside the scope of this paper. Notice that this limitation also affects any in-browser detection system, such as Google CAMP [18], because reporting the file downloads to CAMP could be disabled by the browser exploit code.

An attacker may also attempt to evade the statistical features measured by AMICO. However, while the attacker may be able to evade a few single features, most of AMICO's features are engineered to capture the fundamental characteristics of current evasive behavior of malware download campaigns. Namely, we attempt to capture those characteristics of malware downloads that the attackers already uses to evade existing detection tools, for example by frequently changing domain names, URLs, or the network infrastructure that supports the malware download operations. Therefore, evading the majority of AMICO's features would likely force malware campaigns back into a more "stable" malware distribution infrastructure, which may in turn be more easily blocked by static blacklists, for example. Therefore, we believe AMICO provides a robust complement to existing detection techniques, forcing attackers to incur a significant cost to try to evade both AMICO and current detection tools at the same time.

6 Conclusion

We presented AMICO, a novel system for accurately measuring and detecting malware downloads in live network traffic using *download provenance* information. To this end, AMICO uses a number of statistical features purposely engineering to capture the fundamental characteristics of malware distribution campaigns. We showed that AMICO is able to accurately detect malware downloads with up to 90% true positives at a false positives rate of 0.1%, including many zero-day malware, thus complementing current malware detection tools.

Acknowledgments

This work would not have been possible without the help of Christopher Workman, Jesse Bowling, Charles Leggett, and Alexander Merck, who assisted us throughout the deployment of AMICO on our campus network. We also thank the anonymous reviewers for their helpful comments. This material is based in part upon work supported by the National Science Foundation under Grants No. CNS-1149051 and OCI-1127195. 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.

References

1. Virustotal. `https://www.virustotal.com`.
2. Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a dynamic reputation system for dns. In *Proceedings of the 19th USENIX conference on Security*, USENIX Security'10, 2010.
3. Leyla Bilge, Engin Kirda, Christopher Kruegel, and Marco Balduzzi. Exposure: Finding malicious domains using passive dns analysis. In *Proceedings of Annual Network and Distributed System Security Symposium (NDSS)*, 2011.
4. Leo Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, 2001.
5. Juan Caballero, Chris Grier, Christian Kreibich, and Vern Paxson. Measuring pay-per-install: the commoditization of malware distribution. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, 2011.
6. Marco Cova, Corrado Leita, Olivier Thonnard, Angelos D. Keromytis, and Marc Dacier. An analysis of rogue av campaigns. In *Proceedings of the 13th international conference on Recent advances in intrusion detection*, RAID'10, 2010.
7. Charlie Curtsinger, Benjamin Livshits, Benjamin Zorn, and Christian Seifert. Zozzle: fast and precise in-browser javascript malware detection. In *Proceedings of the 20th USENIX conference on Security*, SEC'11, 2011.
8. Google. Google safe browsing API. `https://developers.google.com/safe-browsing/`.
9. J. Zico Kolter and Marcus A. Maloof. Learning to detect and classify malicious executables in the wild. *J. Mach. Learn. Res.*, 7:2721–2744, 2006.
10. Long Lu, Vinod Yegneswaran, Phillip Porras, and Wenke Lee. Blade: an attack-agnostic approach for preventing drive-by malware infections. In *Proceedings of the 17th ACM conference on Computer and communications security*, CCS '10, 2010.
11. Microsoft. Microsoft PE and COFF specification. `http://msdn.microsoft.com/library/windows/hardware/gg463125`.
12. Microsoft. Smartscreen application reputation - building reputation. `http://blogs.msdn.com/b/ie/archive/2011/03/22/smartscreen-174-application-reputation-building-reputation.aspx`.
13. Jon Oberheide, Evan Cooke, and Farnam Jahanian. Cloudav: N-version antivirus in the network cloud. In *Proceedings of the 17th conference on Security symposium*, SS'08, 2008.
14. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of packed executables for accurate computer virus detection. *Pattern Recogn. Lett.*, 29(14):1941–1946, October 2008.
15. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Mcboost: Boosting scalability in malware collection and analysis using statistical classification of executables. In *Proceedings of the 2008 Annual Computer Security Applications Conference*, ACSAC '08, pages 301–310, 2008.
16. Niels Provos, Panayiotis Mavrommatis, Moheeb Abu Rajab, and Fabian Monrose. All your iframes point to us. In *Proceedings of the 17th conference on Security symposium*, SS'08, 2008.
17. Niels Provos, Dean McNamee, Panayiotis Mavrommatis, Ke Wang, and Nagendra Modadugu. The ghost in the browser analysis of web-based malware. In *Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets*, HotBots'07, pages 4–4, Berkeley, CA, USA, 2007. USENIX Association.
18. Moheeb Abu Rajab, Lucas Ballard, Noé Lutz, Panayiotis Mavrommatis, and Niels Provos. CAMP: Content-agnostic malware protection. In *Proceedings of Annual Network and Distributed System Security Symposium (NDSS)*, February 2013.

19. K. Townsend. R&d: The art of social engineering. *Infosecurity*, 7(4):32–35, 2010.
20. Weka. Weka 3: Data mining software in java. `www.cs.waikato.ac.nz/ml/weka/`.
21. Junjie Zhang, Christian Seifert, Jack W. Stokes, and Wenke Lee. Arrow: Generating signatures to detect drive-by downloads. In *Proceedings of the 20th international conference on World wide web*, WWW '11, 2011.

A List of Features

(a) *Domain Features*

domain_malware_downloads	integer
domain_suspicious_downloads	integer
domain_benign_downloads	integer
domain_total_downloads	integer
domain_malware_ratio	real
domain_suspicious_ratio	real
domain_benign_ratio	real
domain_avg_av_labels	real
domain_avg_trusted_labels	real
domain_unknown_hashes	integer
domain_total_hashes	integer
domain_unknown_hash_ratio	real
2ld_malware_downloads	integer
2ld_suspicious_downloads	integer
2ld_benign_downloads	integer
2ld_total_downloads	integer
2ld_malware_ratio	real
2ld_suspicious_ratio	real
2ld_benign_ratio	real
2ld_avg_av_labels	real
2ld_avg_trusted_labels	real
2ld_unknown_hashes	integer
2ld_total_hashes	integer
2ld_unknown_hash_ratio	real

(b) *Server IP Features*

server_ip_malware_downloads	integer
server_ip_suspicious_downloads	integer
server_ip_benign_downloads	integer
server_ip_total_downloads	integer
server_ip_malware_ratio	real
server_ip_suspicious_ratio	real
server_ip_benign_ratio	real
server_ip_avg_av_labels	real
server_ip_avg_trusted_labels	real
server_ip_unknown_hashes	integer
server_ip_total_hashes	integer
server_ip_unknown_hash_ratio	real
bgp_malware_downloads	integer
bgp_suspicious_downloads	integer
bgp_benign_downloads	integer
bgp_total_downloads	integer
bgp_malware_ratio	real
bgp_suspicious_ratio	real
bgp_benign_ratio	real
bgp_avg_av_labels	real
bgp_avg_trusted_labels	real
bgp_unknown_hashes	integer
bgp_total_hashes	integer
bgp_unknown_hash_ratio	real

(c) *Past File Downloads*

hash_life_time	integer
num_dumps_with_same_hash	integer
hash_daily_dump_rate_per_client	real
estimated_clients_with_same_hash	integer

(d) *Download Request Features*

referer_exists	integer
host_name_exists	integer
extension_class	string
url_length	integer
directory_depth	integer

(e) *URL Features*

url_malware_downloads	integer
url_total_downloads	integer
url_distinct_sha1s	integer
url_struct_malware_downloads	integer
url_struct_total_downloads	integer
url_struct_distinct_sha1s	integer