Gossip: Automatically Identifying Malicious Domains from Mailing List Discussions

Cheng Huang
Sichuan University, China
opcodesec@gmail.com

Shuang Hao
University of California, Santa Barbara, USA
shuanghao@cs.ucsb.edu

Luca Invernizzi
University of California, Santa Barbara, USA
invernizzi@cs.ucsb.edu

Jiayong Liu
Sichuan University, China
jylscu@gmail.com

Yong Fang
Sichuan University, China
yongfangscu@gmail.com

Christopher Kruegel
University of California, Santa Barbara & Lastline, Inc., USA
chris@cs.ucsb.edu

Giovanni Vigna
University of California, Santa Barbara & Lastline, Inc., USA
vigna@cs.ucsb.edu

ABSTRACT

Domain names play a critical role in cybercrime, because they identify hosts that serve malicious content (such as malware, Trojan binaries, or malicious scripts), operate as command-and-control servers, or carry out some other role in the malicious network infrastructure. To defend against Internet attacks and scams, operators widely use blacklisting to detect and block malicious domain names and IP addresses. Existing blacklists are typically generated by crawling suspicious domains, manually or automatically analyzing malware, and collecting information from honeypots and intrusion detection systems. Unfortunately, such blacklists are difficult to maintain and are often slow to respond to new attacks.

Security experts set up and join mailing lists to discuss and share intelligence information, which provides a better chance to identify emerging malicious activities. In this paper, we design Gossip, a novel approach to automatically detect malicious domains based on the analysis of discussions in technical mailing lists (particularly on security-related topics) by using natural language processing and machine learning techniques. We identify a set of effective features extracted from email threads, users participating in the discussions, and content keywords, to infer malicious domains from mailing lists, without the need to actually crawl the suspect websites. Our result shows that Gossip achieves high detection accuracy. Moreover, the detection from our system is often days or weeks earlier than existing public blacklists.

ASIA CCS '17, April 02-06, 2017, Abu Dhabi, United Arab Emirates
© 2017 ACM. ISBN 978-1-4503-4944-4/17/04...$15.00
DOI: http://dx.doi.org/10.1145/3052973.3053017

Keywords

Malware Detection; Natural Language Processing; Blacklists

1. INTRODUCTION

The Internet has grown quickly and provides a wide range of services. However, it has also become an attractive target for cybercriminals who perform malicious activities, such as spam advertising, financial fraud, and drive-by download attacks. These attacks are usually associated with domain names, pointing to the hosts that either contain malicious content or operate as command-and-control servers. When servers are identified as malicious, law enforcement and security experts take them down, but miscreants keep registering new domains to support illicit activities. Security companies and researchers have developed a number of blacklists, such as Spamhaus [5] or PhishTank [4], to detect and filter the domains used in attacks. Existing systems generate blacklists by analyzing network traffic, crawling suspicious domains, or deploying honeypots to identify illicit activities. Although blacklisting is one of the most widely used methods to detect potential attacks, it suffers from two main problems: limited coverage and detection delay. Recent studies have shown that conventional blacklists only capture a small portion of the malicious domains [30, 35], and they experience considerable delay in capturing the malicious domains after the attacks happen [39, 43].

At the early stages of the attacks (i.e., before blacklists successfully report malicious domains), network administrators or security experts might already notice abnormal activities, and they tend to find efficient ways to confirm their observation and exchange findings. Web forums or mailing lists are an ideal place to discuss and share such information. After a user posts a domain in question, other people can reply and provide additional evidence, which helps in deciding whether the concerned domain is indeed malicious. Such online discussion offers several advantages. (1) The observations from many vantage points are combined to reach a conclusion. (2) The knowledge and experience of the security professionals participating in the discussions helps to accurately identify potential attacks. (3) Since these discussions mostly focus on malicious activities that appear in early stages, only a few victims are under attack. If one could build a detection system that uses this information and generates a timely blacklist, it could help to prevent the larger-scale spreading of the attacks. However, the posted contents in mailing lists are unstructured, and human effort is required to uncover useful information. To date, little research has been performed to automatically recognize threats discussed in mailing lists, even though linguistic analysis has been successfully applied to other security-related problems, such as using stylometry to identify anonymous bloggers [37].

In this work, we design and implement Gossip, a detection system to automatically extract malicious domains from the discussions in mailing lists. The main intuition is that these discussions among security experts contribute to characterize the domain names involved in illicit activities. Gossip relies on the features of linguistic patterns, email thread information, and users participating in the discussions. Such features can be easily extracted with small overhead (compared to website crawling or malicious software analysis). We use machine learning techniques to build a statistical classification model, and evaluate the performance on over 40 mailing lists. Our result shows that Gossip achieves high detection accuracy. Moreover, the detection from Gossip is often days or weeks earlier than existing public blacklists; in particular, 40% of the domains could be detected 10 days before their appearance on blacklists. We manually verify the prediction results, and find hundreds of malicious domains that could not be identified by other blacklists. Network administrators can use the output of our system to find attack traffic in their networks. Gossip can effectively complement other detection methods and improve the defense against malicious domains.

In summary, we make the following contributions:

- We develop Gossip, a novel detection system that uses linguistic and machine learning techniques to automatically identify malicious domains based on mailing list discussions.

- We identify four new families of features to leverage the discussions and opinions of security experts on the suspect domains. (1) *URL features* capture the patterns of the embedded URLs; (2) *Thread features* are based on the meta-data associated with the multiple emails in a thread; (3) *Participant features* focus on the participants involved in the email thread discussion; and (4) *Contextual features* examine the text around the domain under analysis. Gossip focuses on the data within the mailing lists and requires little auxiliary information to derive these features.

- We evaluate Gossip on real-world mailing lists. The ground truth information is collected from both public blacklists and manual checking. We show that Gossip is able to identify malicious domains accurately, resulting in a 94% detection rate with zero false positive rate. Our results show that Gossip provides earlier detection compared to existing blacklists.

2. SYSTEM DESIGN

In this section, we introduce the design and the internal details of our system. Gossip aims to quickly and accurately find malicious domains from mailing lists that discuss security-related topics. There are two main challenges that we face. First, the data and information in the mailing lists does not have a fixed format and the content text is not structured. Second, it is difficult to identify effective features which indicate malicious domains. To achieve our goal, we develop components to parse the raw data, extract a set of salient features, and build a model to classify whether the domains are involved in malicious activities. Figure 1 shows the architecture of our system. The input is the email messages from mailing lists. It is noteworthy that Gossip can be easily extended to work on the messages collected from other platforms, such as newsgroups or web forums. The output is a list of domains with classification scores, where higher scores indicate that the domains are more likely to be malicious. Gossip consists of four components: preprocessor, domain extractor, feature parser, and automatic classifier. Next we describe the functionality and implementation of each component in more detail.

2.1 Preprocessor

As the first step, we need to process the raw input data. The messages from a mailing list are stored in the mbox format (RFC 4155) in a single file. We develop an ***email parser*** to separate the mailbox contents into individual email messages in MIME format (RFC 5322) and extract the message attributes, including the header information and the message body. In particular, four header fields contribute to the feature extraction later (see Section 3): the "From" field indicates the author of the message, the "Organization" field refers to the company or organization that the author is affiliated with, the "Content-Disposition" field shows whether the email has attachments, and the "Subject" field contains words to represent the email topic. The "Date" field shows the time when the message was sent, which allows us to compare the discussion time in the mailing list with the time when the domains were eventually included in other blacklists (see Section 4.2.2).

The message body not only contains clean text, but also can include HTML elements or special encoded characters. The process of ***text sanitization*** extracts the plain text from the rendered HTML and removes special characters. We discard the messages that use encryption (e.g., PGP encryption), since the message content is not accessible. Moreover, the parts of the text with little information for our detection (such as user signatures) are excluded from further analysis. For simplicity, we filter out the email messages if they are in non-English languages.

When an email is posted on the mailing list, other people can reply and participate in the discussion, which forms a conversation thread. We develop a module that performs ***thread grouping*** to associate emails belonging to the same conversation. Each email has a unique "Message-ID". A reply message has the "In-Reply-To" field referring to the "Message-ID" of the previous message that it replies to (the first email in a thread has an empty "In-Reply-To" field). Following this process, we link the emails in the same thread based on the header fields "In-Reply-To" and "Message-ID". The output of the preprocessor component is the attributes

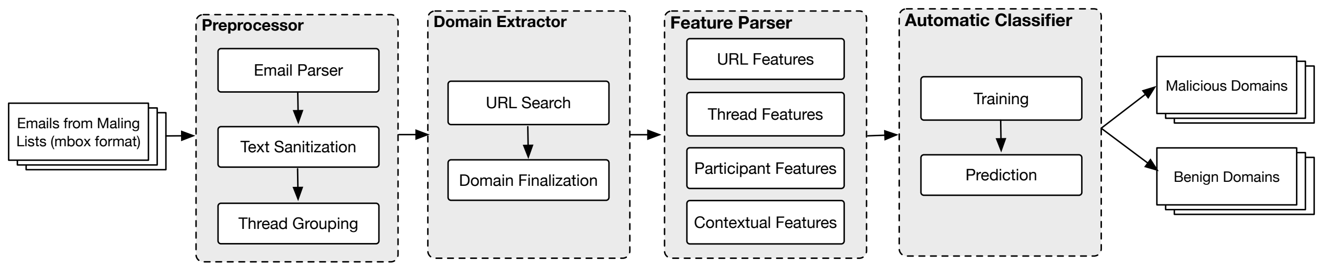

Figure 1: The architecture of Gossip. The system contains four components (with various internal modules).

of email messages (including headers and body text) and the associated threads.

2.2 Domain Extractor

Our analysis target are the domains discussed in the mailing lists, which correspond to parts of the URL strings. We first find URLs in the message bodies, and then derive the domain names. Though at a first glance it might seem trivial to extract URLs from the email text, arbitrary conventions or variants adapted by users often deform the standard URL format and introduce additional difficulty. Since the suspect URLs have high risk of leading to websites that host malicious contents, security professionals tend to substitute or insert characters in the URLs to avoid accidental clicks. For example, a URL `http://www.malwareexample.com/index.html` can be obfuscated as `hxxp//www[.]malwareexample[.]com/index.html`, to make it infeasible to directly click on it. We manually construct regular expression signatures to match the patterns and develop a ***URL search*** module to derive the correct URLs.

We subsequently parse the URLs in the ***domain finalization*** module to obtain the candidate domains. In our analysis, we mainly focus on the second-level domains. For example, from the above URL `http://www.malwareexample.com/index.html`, the extracted domain is `malwareexample.com`. Some country code second-level domains, such as `.com.cn` and `.co.uk`, provide subdomain registrations (as defined in the Mozilla Public Suffix List [3]) to the public. For these domains, we extract the third-level domain names as the analysis target. A domain can appear in multiple threads, so we associate each domain with all the email threads in which it is mentioned for further processing.

2.3 Feature Parser

After the previous steps, each domain is correlated with a set of discussion threads and the corresponding email messages. A key question is: what features best characterize a malicious domain? We explore the properties of the discussion behavior and content in the mailing lists. We group the features into four categories. ***URL features*** examine the lexical patterns of the URLs and the auxiliary information about the domains, such as whether the URL is typed as in the standard scheme or whether the domain is in the Alexa top list. ***Thread features*** focus on the attributes of the threads. We characterize how the domains are discussed, such as how many emails contribute to the discussion or whether the thread contains hash values (often used to fingerprint malware). ***Participant features*** investigate who participates in the discussions. After someone posts questions about suspect domains, other people reply to the message to provide answers or additional observations. The reputation of the participants indicates to what degree people can trust the discussion and conclude that the domain is problematic. ***Contextual features*** consider the semantic patterns of the discussion content, particularly based on the text that surrounds the analyzed domains. It is noteworthy that Gossip only uses the features derived from the discussion threads, and does not need to actively scan the suspect domains or URLs. In Section 3, we explain the features in more detail, and we discuss how to represent them.

To avoid the bias caused by different feature scales, we apply a normalization process on the features. For each feature value $f_n \in F_n$, the normalized value is calculated as in Equation 1, where $max(F_n)$ and $min(F_n)$ correspond to the maximum and minimum values in F_n respectively. The min-max normalization converts the feature values in the range of $[0, 1]$.

$$Normalize(f_n) = \frac{f_n - min(F_n)}{max(F_n) - min(F_n)} \qquad (1)$$

2.4 Automatic Classifier

We incorporate the features into a classification approach to determine whether the domains in discussions tend to get involved in malicious activities. In our application scenario, the feature matrix is large (about 20,000 features) and sparse (with many zero values). We compare multiple classification methods including Naive Bayes, Support Vector Machine (SVM), and Random Forests [13] with real data in our experiments. In particular, Random Forests, which combines a collection of decision trees to vote for the final result, yields the best accuracy and running performance in our experiments. Therefore, we chose to use Random Forests as the automatic classifier in Gossip. In the ***training*** phase, we use the labeled ground truth (based on both manual checking and existing blacklists, see Section 4) to construct the classification model. In the ***prediction*** phase, the classifier outputs a score in the range of $[0, 1]$ for each domain, where a higher score indicates that the domain is more likely to be malicious. Operators can set appropriate thresholds to achieve tradeoffs between the false positive rates and the detection rates. If a domain has a classification score that is greater than the predefined threshold, Gossip reports the domain as malicious.

3. FEATURES

For each domain under analysis, we first aggregate all email threads that mention it. We extract a vector of features from these emails, which we then feed into our machine learning component. In this section, we present each feature category in detail.

3.1 URL Features

We extract two features from each URL. The first is a popularity feature, which indicates whether the URL's domain name is included in the Alexa top one Million domains. This feature introduces apriori knowledge on our classification, for which we tend to classify very popular domains as benign, before looking at the content of email threads.

The second feature indicates whether the URL is likely to lead to a binary download, which is implemented by checking whether the URL contains a ".exe" extension. URLs leading to a binary download are more likely to be malicious.

3.2 Thread Features

We identify all email threads mentioning each URL under analysis, and we extract features from the collection of threads.

Terms used in email subjects. The subject of an email often provides a concise overview of the email content. Because of this, we build a bag-of-words binary vector representing the terms used in these subjects, after stripping common words (e.g., stop words, "Re:",. . .) and punctuation characters. For example, the presence of the words "mailing list memberships reminder" is indicative of benign content. The email subject is extracted from the "Subject" field in the message header.

Number of replies. We indicate in a single feature the total length of the threads where a URL was mentioned. A URL appearing with high frequency is often part of a participant's signature, which indicates it is benign. On the other end of the spectrum, less frequent URLs are more likely malicious. As we discussed, the reply number is calculated based on the "Message-ID" and "In-Reply-To" fields.

Number of domains. We compute the number of distinct domains that co-occur with each URL. This feature is indicative of the presence of "URL dumps", where some emails contain lists of domains that have been used or victimized by the same attacker.

Number of IP addresses. Similarly to the previous feature, we compute the total number of IPv4 addresses that co-occur with each URL. Long lists of IP address are typically used to identify compromised hosts that are under the control of the attacker. We parse the message contents and extract the strings in quad-dotted format (each integer ranging from 0 to 255) as the IP addresses.

Number of attachments. When discussing a particular piece of malware, some participants include additional evidence as attachments. These attachments include malware samples, network dumps, or lists of affected users. We include a feature that counts them. The attachment is indicated by the "Content-Disposition" header field in each email.

Number of organizations. Some companies configure their SMTP servers to include a non-standard email header "Organization" in all outgoing mail. This field is sometime spelled in its plural form, and/or British variant. We encode the name of the organization in a binary vector, which we add to our features. In particular, we generate a one-hot vector for each email, and we aggregate them all in a single vector, that will have a non-zero value for each organization that took part in the discussion.

Number of known malware-detection services. We compute the number of URL, co-located with the URLs under analysis, which point to known malware-detection services (e.g., VirusTotal [20]). To do so, we refer to a short list of these services that we have manually compiled.

Number of cryptographic digests. We compute the number of SHA1 and MD5 digests co-located with each URL under analysis, as these digests are routinely used to uniquely identify malware samples. We leverage the fact that SHA1 and MD5 digests correspond to 40 and 32 hexadecimal numbers respectively, and count the strings matching the criteria.

3.3 Participant Features

The members participating in the mailing lists have different experience, resources, and technical background. Therefore, their contribution to the discussions may vary. The response from some people (especially security experts) is more likely to indicate attacks or malicious activities. We leverage the users' activities and examine who participated in a conversation thread. The identities of the participants could be easily distinguished by their email addresses.

First, we extract all the senders' email addresses from the data sets and compute their hash with the MD5 algorithm in order to avoid disclosing any private information. Second, we use these strings as participant features: If an email address shows to reply a thread, the value will be set to binary value 1. Finally, we generate a mapping matrix to indicate which members have participated in the discussions about the domains.

3.4 Contextual Features

In this last group of features, we extract traits of the text that surrounds each URL under analysis. A challenge is how to find keywords to accurately indicate the URL's reputation. Traditional bag-of-words method based on the whole text or paragraph have several limitations, such as including many common words or sentences not closely related with the subject. Previous research only focuses on analyzing the linguistic patterns in the URL itself [18, 48].

We develop an approach to generate features from the contextual keywords. The heuristic is based on the intuition that the neighbor words close to the discussed URL could represent the judgment of human analysts on the URL.

We build a bag-of-words feature vector from a windowed portion of the text (up to a maximum of 20 words, as a recent study showed that an English sentence typically contains less than 20 words [17, 21]), centered on the URL. If plenty of negative-meaning words are associated with the domain, the domain is more likely to be malicious. For example, consider the following snippet from real-word data:

> The first-stage backdoor is a freshly compiled, slightly modified version of the same one used in the IE 0day phishing waves, and the re-use of a c2 domain **xxxxx.com** is compelling. And the

domain is associated with some malicious activities.

From the text description, a human user can infer that the domain "xxxxx.com" is malicious. We expect Gossip to automatically extract the words that can represent the discussion opinions on the domain. A series of keywords from this paragraph could indicate the attributes or activity information about the domain. In order to select representative words to reflect the domain's reputation, we first remove stop words (which are frequently used common words and do not have any meanings in the sentence) close to the entity's position, select a certain area to limit the number of words, and extract the words that best indicate the maliciousness of the domain. An output result for the above sample is shown below:

> ~~The~~ first-stage backdoor ~~is a~~ fresh~~ly~~ compil~~ed~~, slight~~ly~~ modif~~ied~~ version ~~of the same one used in the~~ IE 0day phish~~ing~~ wav~~es~~, ~~and the~~ re-use ~~of a~~ c2 ~~domain~~ **xxxxx.com** ~~is~~ compelling. ~~And the~~ domain ~~is~~ associat~~ed~~ ~~with some~~ malici~~ous~~ activit~~ies~~.

To automatically extract the contextual features from the email threads, we use four steps: text cleaning, removing stop words, stemming, and extracting contextual words. Next we introduce each step in detail.

Text cleaning. There are various kinds of email formats and encoding methods in addition to plain text (such as HTML pages, non-ASCII characters, etc.). The "Content-Type" in the email header indicates the content type of the message. The encode method could be found in the "Content-Transfer-Encoding" field. In order to correctly extract the content and remove irrelevant HTML codes, we parse the HTML tags with the Beautiful Soup Python library [41] and decode the Base64 encoded content. Moreover, some punctuations are vital to represent the meaning of sentences and show the attitude of the discussions. For example, some punctuations, like "!", "?", could affect the meaning of the sentence, while some other punctuations, like ":", "–", do not represent a user's strong sentiments. In order to build a representative feature set, we construct a punctuation list and include these punctuations into word tokenization.

Removing stop words. We first checked the stop words from the NLTK library [44], but the library only contains a limited set of 127 English words. Google research team has calculated and published the 10,000 most common English words (ordered by their frequencies) [19]. We combined the top 2,000 English words from the Google list and all the stop words from the NLTK library, and then manually removed the words which have sentimental meanings. Eventually, we obtained 1,550 stop words and exclude them in our analysis.

Stemming. Stemming is the process of getting the root form of other-tense-format verbs (such as present tense, past tense, or future tense). In our work, we treat different tense-format words as one feature, and use the Porter stemming module [49] to convert each word into its root form (which is less aggressive than other stemming algorithms [22, 40]).

Extracting contextual words. To get contextual information about a domain under analysis, we extract text tokens from all the threads in which the domain is mentioned. While a naive choice is to select all tokens showing in the threads, the approach would confuse the classifier: Users may discuss both benign and malicious domains in the same thread. Using a bag-of-words approach at the thread level would yield the inclusion of irrelevant or even misleading word tokens for the analyzed domain. The key observation to resolve this issue is that the tokens in the immediate vicinity of each analyzed domain best indicate the potential maliciousness of the domain. Therefore, for each domain we only extract a subset of text tokens close to its position in the text.

4. EVALUATION

We evaluate the performance of Gossip on real-world mailing list data. First, we describe how we collected our datasets and obtained ground truth. We then present experimental results regarding the detection accuracy and timing compared with existing blacklists.

4.1 Datasets

The mailing list data that we use include two types of lists: security mailing lists that focus on discussing malicious activities and attacks, and public mailing lists that discuss general computer-related issues. These mailing lists provide real-world examples of people taking advantage of online forums to share information and find solutions. We collected a number of blacklists to label the domains and use them to evaluate the performance of our system.

Security mailing lists. Security professionals from different organizations or enterprises establish mailing lists to exchange observations and security opinions. Since such mailing lists might contain private information about network configurations or company security products, the content is usually not accessible to the public and the subscription requests need moderator's approval. We subscribed to six security mailing lists and collected the email discussions from June 2009 to May 2015. These mailing lists focus on security-related topics, and post questions about suspect domains. After processing the raw data, we extracted 10,780 email threads and 2,649 distinct domains.

Public mailing lists. We collected data from public mailing lists to further examine the robustness of our system. The public mailing lists provide a platform where people can discuss general computer-related issues. It is unlikely that the discussion contents contain malicious domains, so we use the public mailing lists as a control group to make sure that Gossip has little misclassification on benign domains (see Section 4.3). A collection of mailing lists can be found at the "MARC" archive (marc.info), which stores 70 million emails across over 3,500 mailing lists from seven million users. We randomly sampled 43 mailing lists and crawled 76,380 email threads from November 1999 to September 2015, which contain 3,898 distinct domain names.

Blacklist labels. We crawl data from public blacklists and online malware detection services to obtain ground truth information. It is noteworthy that existing blacklists do not provide perfect ground truth, so we further performed manual validation on the detection results of Gossip (see Section 4.2.1). Some blacklist services have timestamp information to indicate when the domains were blacklisted, which allows us to compare the time difference between the de-

Table 1: Statistics of 244 sampled domains in security mailing lists (as training data).

Category	Count and percentage
Malicious domains	93 (38.11%)
Benign domains	141 (57.79%)
Uncertain domains	10 (4.10%)

tection of existing blacklists and our system. Note that in our experiments we only collected and compared with public blacklists. While commercial blacklists contain more timely information, Gossip helps to automatically extract threats from the online discussions of human analysts. The blacklists that we use are from three main groups.

- Feed of multiple blacklists. We obtained a feed which collected over 75 different public blacklists [29], including malwaredomainlist [2], abuse.ch [6], Malc0de [1], Spamhaus [5], and PhishTank [4]. The collection period spans from July 2011 to June 2015, and the data covers domain names involved in various malicious activities, such as phishing, spam, and drive-by download attacks. Given a domain name, the feed shows the particular blacklists that reported the domain as malicious and the corresponding blacklisting periods.

- VirusTotal [20]. To complement the above blacklist feeds, we query VirusTotal, which integrates over 60 different security products. However, VirusTotal produces a non-negligible number of false positives [45] and, therefore, the domains in our experiment are labeled as malicious only if more than one security product flagged as problematic. VirusTotal provides timestamps to indicate the first time a domain was analyzed.

- McAfee SiteAdvisor [33]. Another resource is SiteAdvisor, a free online service from McAfee to check whether a domain is associated with malicious activities. We submitted the domains that have been found from the mailing lists to the McAfee SiteAdvisor. Although the service returns many labels for malicious domains, a deficiency is that it does not provide time stamps associated with the detection.

In our experiments, we consider a domain as malicious if any of the above blacklists has reported that the domain was associated with illicit activities.

4.2 Experiments on Security Mailing Lists

We first examine the performance of Gossip on the security mailing lists, and compare the results with existing blacklists. One challenge is that it is uncertain which domains discussed in the security mailing lists are the benign ones. So we sample a set of domains and manually check whether they are malicious. To make sure our samples are representative, we do not pick more than one domain from the same email thread (which have similar features). Table 1 shows the manual checking results of the sampled domains. We use 93 manually checked malicious domains and 141 benign domains as training data.

To derive unbiased testing data, we remove the domains appearing in the same discussion threads as those in the training data. In other words, we separate the training data and testing data and make them unassociated. We obtain 2,405 domains from the security mailing lists as the testing dataset.

4.2.1 *Accuracy and Manual Validation*

We use the algorithm of Random Forests on the training data to learn a classification model, and apply the model on the testing data. The output result of each domain is a score, where a higher score indicates that the domain is more likely to be malicious. The domains that Gossip classifies as malicious depend on the selected score threshold. In our experiments, we set a 0.4 threshold (the value is in the range of $[0, 1]$), which results in detecting most of the blacklisted domains. There are 1,559 malicious domains being classified by Gossip as malicious. The blacklists reported 693 malicious domains discussed in the mailing lists, among which 608 domains (88.7%) are also captured by Gossip. The result shows that Gossip can cover most of the malicious domains detected by existing blacklists. Recall that we do not have a perfect ground truth about malicious and benign domains. To further examine how accurate our results are, we sorted the domains in descending order according to the detection scores, and manually investigated the top 200 domains, the 100 domains with scores right above 0.4 threshold (ranking 1,460–1,559), and the last 100 domains (i.e., those close to the end of 2,405 domains in list). Table 2 shows the count of blacklisted domains (the second column) and manual checking results of the non-blacklisted domains (the third to the fifth columns). We see that the domains with top classification scores contain considerable amount of blacklisted domains: the first 100 domains have 60 blacklisted, and the next 100 domains have 62 blacklisted. When we manually investigated the non-blacklisted domains by checking the web content of the domains and the evidence in the mailing list discussions, in the 1–100 and 101–200 groups, we found that 90.0% and 76.4% of the non-blacklisted domains were actually malicious ones, but missed by existing blacklists. Among the 100 domains with ranking 1,460–1,559, there were still a substantial amount of malicious domains that were not blacklisted. The observation demonstrates that Gossip complements the existing blacklists, and captures more malicious domains. In Section 5.1, we further analyze the false positives and show that whitelisting can significantly improve the accuracy of Gossip. The last row in the table shows the 100 domains with the lowest classification scores, which are supposed to be benign domains. Few of them appear in blacklists or are determined to be malicious via manual checking. The results demonstrate that our system can effectively distinguish malicious domains from benign ones.

4.2.2 *Comparison of Detection Timing*

Detection time is an important metric for blacklists and detection systems, since early detection can prevent malicious activities from spreading to more victims. We investigate and compare when the malicious domains appeared on blacklists and when they could be detected by Gossip. As we mentioned, McAfee SiteAdvisor does not provide timestamps in the data, and therefore, we exclude it from the analysis. With the 0.4 threshold above, we have 476 malicious domains being flagged by both reputation services (excluding McAfee SiteAdvisor) and Gossip.

We use the timestamp (in the "Date" header field) of the last email of the thread that discussed a domain as the time

Table 2: Results of manual checking on domains that Gossip classifies as malicious (ranked by the classification scores). In each row, the combination of the numbers in bold is the ground truth of malicious domains.

Rank region	Blacklisted	Non-blacklisted Malicious	Benign	Uncertain
1–100	**60**	**36** (90%)	4 (10%)	0 (0%)
101–200	**62**	**29** (76%)	8 (21%)	1 (3%)
...	...	...	...	...
1,460–1,559	**20**	**22** (27%)	55 (69%)	3 (4%)
...	...	...	...	...
2,306–2,405	**4**	**0** (0%)	96 (100%)	0 (0%)

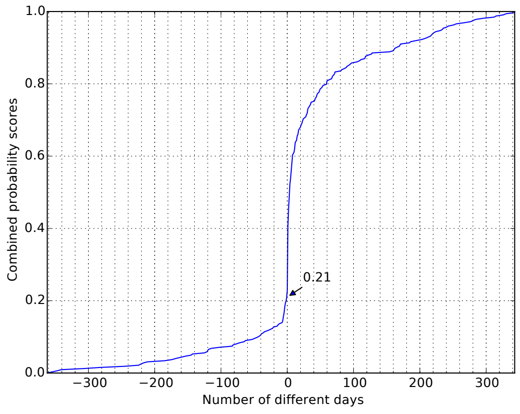

Figure 2: Distribution of the detection time difference between Gossip and public blacklists (one year period).

when Gossip could detect the domain. Each blacklist provides the first time and last time when it flagged a domain as malicious, which forms a blacklisting window. Since different blacklists could generate multiple blacklisting windows, we need to find the appropriate window to compare with Gossip. The detailed process of calculating the time difference is shown in Algorithm 1.

Figure 2 presents the distribution of the days between blacklisting time and when Gossip made the detection. We only show the time difference within one year, since it is unlikely that the gap is over one year (a domain might expire after one year and change ownership). The positive values on the x-axis represent that Gossip could detect the malicious domains earlier than blacklists. We see that 79% of the malicious domains were detected sooner by Gossip than existing public blacklists. Figure 3 shows the truncated distribution within 30 days (one month). About 40% of the domains could be detected 10 days before they appeared on public blacklists. Therefore, Gossip has a better chance to detect attacks early and provide more time for users to mitigate the threats.

4.3 Experiments on Public Mailing Lists

In order to evaluate the effectiveness of Gossip, we combine the public mailing lists to construct a synthetic dataset. The public mailing lists usually do not contain malicious domains, since their topics are about general computer-related issues (not focusing on security or attacks). We extract the domains discussed in the public mailing lists as the benign set, and remove any domains that have appeared in blacklists. We use the blacklisted domains in the security mailing lists as the malicious set. At the end, we have 693 malicious domains and 3,898 benign domains.

Most of the features can be extracted as previously. One special case is participant features. It is unlikely that the security mailing lists and the public mailing lists have overlapping users (with the same email addresses). Therefore, we extend the participant features as the second-level domain parts in the email addresses (instead of using the whole email addresses as the feature). We use 10-fold cross validation to evaluate the detection accuracy. The domains are randomly partitioned into 10 equal-size parts. At each iteration, one single part is taken as the testing data, and the other parts are used as training data. The final result is the average over all iterations. As discussed before, we use Random Forests as the classifier.

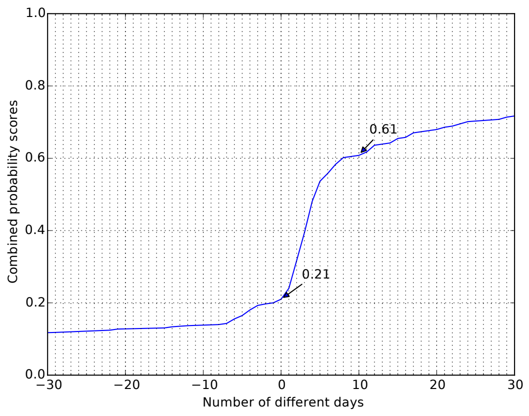

Figure 3: Distribution of the detection time difference between Gossip and public blacklists (truncated, one month period).

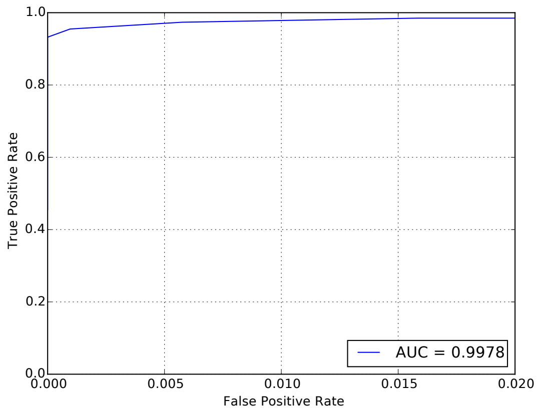

Figure 4: ROC curve of Gossip (10-fold cross validation).

Algorithm 1 Calculating the time difference between blacklisting and GOSSIP detection.

```
Require:
    $d$: target domain;
    $BT$: set of appear time period in multiple-blacklist feed (MD);
    $vt$: first seen time in VirusTotal;
    $mt$: first seen time in mailing list;
    $dt$: different time for domain;
 1: $BT \Leftarrow$ search all appear time in MD for $d$
 2: if $BT \neq$ empty then
 3:     if exist($ibt \in BT$) &
        $ibt.first.time \leq mt \leq ibt.last.time$ then
 4:         $dt \Leftarrow min(ibt.first.time - mt)$
 5:     else if exist($ibt \in BT$) &
        $ibt.first.time \geq mt$ then
 6:         $dt \Leftarrow min(ibt.first.time - mt)$
 7:     else if exist($ibt \in BT$) &
        $ibt.first.time \leq mt$ then
 8:         $dt \Leftarrow min(ibt.first.time - mt)$
 9: else if $d \in vt$ then
10:     $dt \Leftarrow vt - mt$
11: return $dt$
```

Figure 4 shows the ROC curve of the 10-fold cross validation. GOSSIP achieves high accuracy, namely a 94% detection rate with zero false positives. The area under curve (AUC) in the figure is about 0.99. Our result shows that GOSSIP can be deployed across mailing lists and effectively distinguish malicious domains based on the online discussions.

5. ANALYSIS AND MEASUREMENT

Based on the results from the previous section, we perform further analysis to investigate the false positives, evaluate what are the most important features, and characterize the types of the malicious domains that we found in the mailing list discussions.

5.1 False Positive Analysis

In Section 4.2.1, we see that even though GOSSIP is able to extract many malicious domains from security mailing lists, it also classifies a certain amount of benign domains as malicious (i.e., false positives). We investigated the false positives and found that most of them were security blogs or company websites. The reason is that during the discussions about malicious activities, security professionals often include external blogs or technical websites as references to find more details. These benign domains shared similar features in our system as the discussed malicious domains, and therefore had a certain probability to be misclassified. If we can whitelist these security-related domains, the performance of GOSSIP can be further improved. A heuristic is that malicious websites would not show security-related words, such as "malware" or "threat", to alert the visitors. On the other hand, the security-related websites often contain such words when mentioning the malicious activities. Therefore, we developed a bag-of-words model to identify the keywords on the security-related websites, and then crawled the domains in our dataset to whitelist the benign ones.

Table 3: Selected security websites as the seed to extract security-related words.

No.	URL of the website
1	http://lastline.com
2	https://usa.kaspersky.com
3	https://www.fireeye.com
4	https://www.damballa.com
5	http://home.f-secure.com
6	https://www.lancope.com
7	https://www.alienvault.com
8	http://www.avg.com
9	http://www.trendmicro.com
10	https://www.paloaltonetworks.com

We selected 10 well-known security websites (listed in Table 3) as the seed to extract security-related words. Algorithm 2 explains the detailed extraction steps using natural language processing technology to generate security keywords (line 1–8) and ranking these keywords by term frequency (TF) and the inverse document frequency (IDF) algorithm (line 9–14). To find what keywords best represent security topics, we calculated the TF-ID value for each word, where the IDF value was calculated from Microsoft's N-gram service [47]. We ranked the words according to the TF-IDF values, and selected the top 50 words to generate a security keyword corpus.

The top 50 security-related keywords in our experiment are: "security", "threat", "amp", "avg", "kaspersky", "lastline", "business", "protection", "cyber", "2015", "endpoint", "network", "advanced", "intelligence", "learn", "alienvault", "malware", "products", "trend", "enterprise", "get", "threats", "antivirus", "mobile", "partner", "fireeye", "internet", "free", "support", "data", "micro", "damballa", "detection", "management", "home", "small", "attacks", "lancope", "detect", "contact", "new", "incident", "defense", "online", "united", "delivers", "breach", "zeroday", "virus", "services".

Next, we crawled the domains in our dataset and extracted the words from their websites. If a website has at least N words in the security keyword corpus, we think the corresponding domain is a security-related domain (applicable for whitelisting). We set N as 5 in our experiments.

We re-visited the manual checking process on the GOSSIP's output. Table 4 shows the result with whitelisting, compared to Table 2. The numbers in the parentheses represent the counts of domains that can be filtered by the whitelist. Especially, for the domains with lower ranking, like those in the 1,460–1,559 group, a considerable portion of the misclassified benign domains can be whitelisted. With whitelisting, GOSSIP can achieve higher detection accuracy.

5.2 Feature Importance

With the classification results, we identify what the most effective features in GOSSIP are. This insight can help future feature development, and might stimulate mailing lists or Web forums to provide structured formats or interface to extract the useful features. We evaluated the feature importance by using the mean decrease impurity (MDI) method [38]. Table 5 shows the 10 most effective features in our system. Among the top five features, four of them belong to *URL features* and *Thread features*, which require small overhead to extract.

Algorithm 2 Extraction of keywords from security websites with TF-IDF algorithm

```
Require:
    $D$: set of testing domains;
    $TW$: set of words from all websites;
    $TFIDF$: set of td_idf values for all words;
    $RV$: set of top 50 security keywords;
 1: for each $d \in D$ do
 2:     $W \Leftarrow extract\_text\_from\_html(d)$;
 3:     $W \Leftarrow word\_tokenize(W)$;
 4:     $W \Leftarrow remove\_stopwords(W)$;
 5:     for each $w \in W$ do
 6:         if $len(w) \in [2, 20]$ then
 7:             $w \Leftarrow stemming(w.lower())$;
 8:             $TW.append(w)$;
 9: for each $tw \in TW$ do
10:     $tf \Leftarrow TW.counter(tw)/len(TW)$
11:     $idf \Leftarrow Microsoft\_Ngram\_idf(tw)$
12:     $TFIDF.append((tf * idf, tw), tw)$
13: $TFIDF \Leftarrow order\_by\_value(TFIDF)$;
14: $RV \Leftarrow TFIDF.get\_most\_50\_words$;
15: return $RV$;
```

Table 4: Results of whitelisting compared to Table 2. The numbers in parentehses indicate the amount of whitelisted domains.

Rank		Non-blacklisted		
region	Blacklisted	Malicious	Benign	Uncertain
1–100	60	36	4 (**-2**)	0
101–200	62	29	8 (**-3**)	1 (**-1**)
...	...	...	...	...
1,460–1,559	20	22 (**-4**)	55 (**-31**)	3 (**-1**)
...	...	...	...	...
2,306–2,405	4	0	96 (**-48**)	0

To show a complete picture about the feature ranking, we list the next 40 features. The keywords in the *Contextual features* are put in quotes. Since the *Participant features* contain personal information, we just show the feature indexes. The ranked feature list is: “fw”, “folk”, “server”, “more”, “associate”, number of known malware-detection services, “intel”, “hash”, “hxxp”, number of domains, “public”, participant feature #456, “reject”, “refer”, “lot”, “korean”, “marco”, “jeroen”, number of IP addresses, “down”, “origin”, “regard”, feature #132, “incident”, “intrude”, “iframe”, participant feature #387, “poc”, participant feature #787, “uri”, “firm”, “advance”, “investigate”, participant feature #1287, “against”, “attack”, “jose”, participant feature #21, “myself”.

5.3 Domain Characteristics in Mailing Lists

We provide high-level statistics of the malicious domains discussed in the mailing lists, to understand what malicious activities draw the most discussions, and where they come from. We manually checked the sampled malicious domains (see Section 4.2.1), and put them into different categories. In Table 6, the first column shows the category of the malicious activities, and the second column is the percentage of the domains in that category over all malicious domains. We see that drive-by download attacks, C&C servers, and malware download servers account for the most significant percentages. Presumably since these attacks are more stealthy

Table 5: The 10 most important features in Gossip.

No.	Feature	Feature group
1	Alexa rank	URL features
2	“n’t”	Contextual features
3	Number of replies	Thread features
4	Number of attachments	Thread features
5	“something”	Contextual features
6	URL ending with .exe	URL features
7	“believe”	Contextual features
8	“not”	Contextual features
9	Number of organizations	Thread features
10	Number of cryptographic digests	Thread features

Table 6: Distribution of the categories of the malicious activities discussed in mailing lists.

Category	Domain percentage
Drive-by download	34.57%
C&C	22.84%
Malware download server	17.28%
Phishing	9.88%
Web backdoor	6.17%
Spam	5.56%
DDoS botnet	3.70%

(i.e., not easy to detect) and cause direct damages, security experts are inclined to discussing them more often in the mailing lists.

Next, we investigate DNS infrastructure and what parent zones host the most malicious domains. Note that in our analysis, some parent zones are top-level domains, and some are second-level domains (as discussed in Section 2.2). Table 7 shows the distribution of the parent zones. The “.com” zone is still the leading target that cybercriminals use for illicit activities. Among the country code top-level domains, the “.ru” zone is the most abused one.

6. LIMITATIONS

As other malware detection systems with machine learning techniques, our approach has some limitations when being applied to mailing lists. In this section, we discuss these limitations, and put forward how Gossip could work more effectively in real-world deployments.

Most of the threads in mailing lists include many discussion on a post, but there are also some threads with very few replies or limited amount of words. If any malicious domain is mentioned in the threads with few replies or words, it is difficult to correctly classify the domain. Such cases contribute to a higher false negative rate. This limitation has been studied in previous research [36]. Stylistic features, such as inferential conjunctions, could help to improve the accuracy of predictions if the amount of training data is large enough, or if one can actively crawl the domains to obtain additional information.

Another issue is that some threads may include a large number of domains and contain few words in the content, e.g., when people discuss many domains resolved to the same IP address. If the training datasets include many such entities, the detection system would be less effective. However, a solution to avoid this issue is to drop such mailing threads

Table 7: Distribution of the parent zones of the malicious domains discussed in mailing lists.

Zone	Domain coverage	Zone	Domain coverage
com	47.66%	com.tw	0.87%
net	7.31%	us	0.87%
org	6.32%	in	0.87%
ru	5.78%	ch	0.76%
info	2.84%	edu	0.65%
de	2.18%	co.uk	0.65%
com.br	1.96%	co.za	0.65%
eu	1.64%	at	0.65%
pl	1.64%	co.kr	0.65%
com.au	1.42%	tk	0.55%
biz	1.31%	es	0.55%
nl	1.09%	Others	11.13%

in the training data sets or just extract a certain amount of domains from the same threads. There also exist other mitigation methods, e.g., previous research extracted more features from URLs (length, digits numbers, etc.) [48], Xiang et al. proposed more features from the Web page and WHOIS information of the entity [50]. But such methods need more network requests and timely response to crawl Web pages.

We use natural language processing techniques to extract contextual features, as in Section 3, but due to the diversity in countries and culture, people may use various words and sentences, and might have different writing errors and grammar mistakes in the threads. Sadia et al. has used stylometric methods to identify anonymous authors by analyzing their writing style [7]. In order to fix these issues, people can adapt more complicated linguistic approaches to extract the contextual features.

7. RELATED WORK

Malware detection is a common problem in security, and has attracted a lot of research interest. Different researchers use various approaches to study this topic, such as dynamic analysis, lexical patterns in the URLs, and the analysis of the underground market. In the following, we highlight the most relevant previous work.

Malware detection. There are several kinds of features to indicate the difference between malicious and benign websites: suspicious code in Web pages, browser behaviors in sandbox, or domain meta-data. These schemes for detecting malicious domains are often leveraged in machine learning techniques. These features can be based on static characteristics of website content [12, 16, 27], or could be extracted from their dynamic behaviors [14, 15, 28]. Domain registration information and DNS traffic also provide important patterns in recognizing malicious domains [9, 10, 11, 23, 24, 25, 51], but such features need to be extracted with external resources or at particular network vantage points. Various machine learning classifiers could be used to classify these entities after the feature extraction.

Blacklisting. There are a number of research works focusing on blacklists: Kührer et al. presented a blacklist parsing system to track 49 different blacklists and provided an analysis of the malicious activities originating from these public blacklists [29]. Kührer found that 15 public blacklists include less than 20% of the malicious domains for a majority of popular malware families [30]. Ma et al. described an approach to classify malicious and benign URLs based on machine learning methods using lexical and host-based features [32]. Almuhimedi et al. studied human behaviors in reaction to the malware warnings in Google Chrome [8]. Such studies have shown that blacklisting has been widely used to block malicious domains and IP addresses, but also outlined several shortages of blacklists. Liao et al. put forward a solution for fully automated Indicators of Compromise (IOC) extraction only based on context terms [31], but mailing list discussions have different situations, in which technical descriptions have fewer words and the same topic has multiple discussions. Thus, their solution could not work.

Natural language processing technologies. NLP technologies have been widely used in security recently, especially in detecting phishing and malicious domains: Darling et al. developed a classification system based on lexical features of URLs [18]. Other researchers also studied the detection of malicious domains using lexical and word segmentation techniques to extract various features such as domain name length, frequencies of certain characters [26, 34, 46]. The N-Gram method also has been used to detect unknown network attacks [42].

To compare with previous research, our approach uses contextual words and machine learning techniques to detect malicious domains in mailing lists. This study could analyze suspicious domains which traditional detection solutions miss, because many malicious domains only appear a few times and last for a very short time. We also propose the *Participant features* based on the historic relationship between domains and security experts. Finally, we use natural language processing technologies to extract features based on the contextual words, which have not been used in this domain before.

8. CONCLUSION

In this paper, we designed Gossip, a novel light-weight detection system to extract malicious domains from mailing lists. We identified four groups of features: URL features, thread features, participant features, and contextual features. We incorporated these features into a classifier to develop a detection system. To derive meaningful keywords for each domain, we introduced an improved continuous bag-of-words model that could extract the representative words around the discussed domains. To filter security company websites or technical blogs in the mailing lists, we proposed an approach to crawl keywords from the Web pages and reduce the false positives. We evaluated Gossip with real-word data. Our results show that Gossip achieves high accuracy with private mailing lists, and captures hundreds of malicious domains that existing public blacklists have missed. Moreover, our system detects 79% of the malicious domains earlier than existing public blacklists. With the experiment on public mailing lists, Gossip gets a 94% detection rate with zero false positives, which also proves the reliability and accuracy of our model.

References

[1] Malc0de database. http://malc0de.com/database.

[2] Malware domain list. http://www.malwaredomainlist.com.

[3] Mozilla public suffic list. http://publicsuffix.org.

[4] Phishtank. https://www.phishtank.com.

[5] The spamhaus project. https://www.spamhaus.org.

[6] The swiss security blog. https://www.abuse.ch.

[7] S. Afroz, A. C. Islam, A. Stolerman, R. Greenstadt, and D. McCoy. Doppelgänger finder: Taking stylometry to the underground. In *IEEE Symposium on Security and Privacy*, 2014.

[8] H. Almuhimedi, A. P. Felt, R. W. Reeder, and S. Consolvo. Your reputation precedes you: History, reputation, and the Chrome malware warning. In *Symposium on Usable Privacy and Security (SOUPS)*, 2014.

[9] M. Antonakakis, R. Perdisci, D. Dagon, W. Lee, and N. Feamster. Building a dynamic reputation system for DNS. In *Proceedings of 19th USENIX Security Symposium*, 2010.

[10] M. Antonakakis, R. Perdisci, W. Lee, N. Vasiloglou II, and D. Dagon. Detecting malware domains at the upper DNS hierarchy. In *Proceedings of 20th USENIX Security Symposium*, 2011.

[11] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. EXPOSURE: Finding malicious domains using passive DNS analysis. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2011.

[12] D. Canali, M. Cova, G. Vigna, and C. Kruegel. Prophiler: A fast filter for the large-scale detection of malicious web pages. In *Proceedings of the International World Wide Web Conference (WWW)*, 2011.

[13] R. Caruana and A. Niculescu-Mizil. An empirical comparison of supervised learning algorithms. In *Proceedings of the 23rd International Conference on Machine Learning*, 2006.

[14] C.-M. Chen, J.-J. Huang, and Y.-H. Ou. Detecting web attacks based on domain statistics. In *Intelligence and Security Informatics*, pages 97–106. Springer, 2013.

[15] M. Cova, C. Kruegel, and G. Vigna. Detection and analysis of drive-by-download attacks and malicious javascript code. In *Proceedings of the World Wide Web Conference (WWW)*, 2010.

[16] C. Curtsinger, B. Livshits, B. Zorn, and C. Seifert. Zozzle: Low-overhead mostly static Javascript malware detection. In *Proceedings of 20th USENIX Security Symposium*, 2011.

[17] M. Cutts. *Oxford guide to plain English*. OUP Oxford, 2013.

[18] M. Darling, G. Heileman, G. Gressel, A. Ashok, and P. Poornachandran. A lexical approach for classifying malicious URLs. In *IEEE International Conference on High Performance Computing & Simulation (HPCS)*, pages 195–202, 2015.

[19] G. groups. 10,000 most common English words. https://github.com/first20hours/google-10000-english.

[20] G. groups. VirusTotal. https://www.virustotal.com.

[21] R. Gunning et al. How to take the fog out of writing. 1964.

[22] N. Habash, O. Rambow, and R. Roth. Mada+ tokan: A toolkit for Arabic tokenization, diacritization, morphological disambiguation, POS tagging, stemming and lemmatization. In *Proceedings of the 2nd International Conference on Arabic Language Resources and Tools (MEDAR)*, 2009.

[23] S. Hao, N. Feamster, and R. Pandrangi. Monitoring the initial DNS behavior of malicious domains. In *Proceedings of the ACM Internet Measurement Conference*, 2011.

[24] S. Hao, A. Kantchelian, B. Miller, V. Paxson, and N. Feamster. Predator: Proactive recognition and elimination of domain abuse at time-of-registration. In *ACM Conference on Computer and Communications Security*, 2016.

[25] S. Hao, M. Thomas, V. Paxson, N. Feamster, C. Kreibich, C. Grier, and S. Hollenbeck. Understanding the domain registration behavior of spammers. In *Proceedings of the ACM Internet Measurement Conference*, 2013.

[26] Y. He, Z. Zhong, S. Krasser, and Y. Tang. Mining DNS for malicious domain registrations. In *Proceedings of the 6th International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2010.

[27] L. Invernizzi, P. M. Comparetti, S. Benvenuti, C. Kruegel, M. Cova, and G. Vigna. EvilSeed: A guided approach to finding malicious web pages. In *IEEE Symposium on Security and Privacy*, 2012.

[28] A. Kapravelos, M. Cova, C. Kruegel, and G. Vigna. Escape from monkey island: Evading high-interaction honeyclients. In *Proceedings of the 8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA)*. 2011.

[29] M. Kührer and T. Holz. An empirical analysis of malware blacklists. *Praxis der Informationsverarbeitung und Kommunikation*, 35(1):11–16, 2012.

[30] M. Kührer, C. Rossow, and T. Holz. Paint it black: Evaluating the effectiveness of malware blacklists. In *Symposium on Recent Advances in Intrusion Detection*. 2014.

[31] X. Liao, K. Yuan, X. Wang, Z. Li, L. Xing, and R. Beyah. Acing the IOC game: Toward automatic discovery and analysis of open-source cyber threat intelligence. In *ACM Conference on Computer and Communications Security*, 2016.

[32] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious web sites from suspicious URLs. In *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2009.

[33] McAfee. https://www.siteadvisor.com.

[34] D. K. McGrath and M. Gupta. Behind phishing: An examination of phisher modi operandi. In *Proceedings of the USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET)*, 2008.

[35] T. Moore and R. Clayton. Evaluating the wisdom of crowds in assessing phishing websites. In *Proceedings of the Conference on Financial Cryptography and Data Security*. 2008.

[36] S. Mukherjee, G. Weikum, and C. Danescu-Niculescu-Mizil. People on drugs: Credibility of user statements in health communities.

In *Proceedings of the 20th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2014.

[37] A. Narayanan, H. Paskov, N. Z. Gong, J. Bethencourt, E. Stefanov, E. C. R. Shin, and D. Song. On the feasibility of internet-scale author identification. In *IEEE Symposium on Security and Privacy*, 2012.

[38] L. Olshen, C. J. Stone, et al. Classification and regression trees. *Wadsworth International Group*, 93(99):101, 1984.

[39] A. Pitsillidis, C. Kanich, G. M. Voelker, K. Levchenko, and S. Savage. Taster's choice: A comparative analysis of spam feeds. In *Proceedings of the ACM Internet Measurement Conference*, 2012.

[40] M. F. Porter. Snowball: A language for stemming algorithms, 2001.

[41] L. Richardson. Beautiful soup documentation. 2007.

[42] K. Rieck and P. Laskov. Detecting unknown network attacks using language models. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. 2006.

[43] S. Sheng, B. Wardman, G. Warner, L. F. Cranor, J. Hong, and C. Zhang. An empirical analysis of phishing blacklists. In *Proceedings of Sixth Conference on Email and Anti-Spam (CEAS)*, 2009.

[44] B. Steven, E. Klein, and E. Loper. Natural language processing with Python. *OReilly Media*, 2009.

[45] P. Vadrevu, B. Rahbarinia, R. Perdisci, K. Li, and M. Antonakakis. Measuring and detecting malware downloads in live network traffic. In *Proceedings of the European Symposium on Research in Computer Security*, 2013.

[46] K. Wang, C. Thrasher, and B.-J. P. Hsu. Web scale NLP: A case study on URL word breaking. In *Proceedings of the 20th International Conference on World Wide Web*, 2011.

[47] K. Wang, C. Thrasher, E. Viegas, X. Li, and B.-j. P. Hsu. An overview of Microsoft Web N-gram corpus and applications. In *Proceedings of the NAACL HLT 2010 Demonstration Session*, pages 45–48. Association for Computational Linguistics, 2010.

[48] W. Wang and K. E. Shirley. Breaking bad: Detecting malicious domains using word segmentation. In *IEEE Web 2.0 Security and Privacy Workshop*. 2015.

[49] P. Willett. The Porter stemming algorithm: then and now. *Program*, 40(3):219–223, 2006.

[50] G. Xiang, J. Hong, C. P. Rose, and L. Cranor. Cantina+: A feature-rich machine learning framework for detecting phishing web sites. *ACM Transactions on Information and System Security (TISSEC)*, 14(2):21, 2011.

[51] W. Zhang, W. Wang, X. Zhang, and H. Shi. Research on privacy protection of WHOIS information in DNS. In *Computer Science and its Applications*, pages 71–76. Springer, 2015.