MalShoot: Shooting Malicious Domains Through Graph Embedding on Passive DNS Data

Chengwei Peng[1,2], Xiaochun Yun[1,3(✉)], Yongzheng Zhang[3], and Shuhao Li[3]

[1] Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing, China
[3] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
{pengchengwei,yunxiaochun,zhangyongzheng,lishuhao}@iie.ac.cn

Abstract. Malicious domains are key components to a variety of illicit online activities. We propose `MalShoot`, a graph embedding technique for detecting malicious domains using passive DNS database. We base its design on the intuition that a group of domains that share similar resolution information would have the same property, namely malicious or benign. `MalShoot` represents every domain as a low-dimensional vector according to its DNS resolution information. It automatically maps the domains that share similar resolution information to similar vectors while unrelated domains to distant vectors. Based on the vectorized representation of each domain, a machine-learning classifier is trained over a labeled dataset and is further applied to detect other malicious domains. We evaluate `MalShoot` using real-world DNS traffic collected from three ISP networks in China over two months. The experimental results show our approach can effectively detect malicious domains with a 96.08% true positive rate and a 0.1% false positive rate. Moreover, `MalShoot` scales well even in large datasets.

Keywords: Domain reputation · Graph embedding · Domain representation · Malicious domains detection

1 Introduction

The Domain Name System (DNS) servers as one of the most fundamental Internet components and provides critical naming services for mapping domain names to IP addresses. Unfortunately, it has been abused by miscreants for various illegal attack campaigns (*e.g.*, directing victims to malicious Web sites [1], exploiting algorithmically generated domains to circumvent the take-down [2,3]). Cisco 2016 annual security report [4] measured that 91.3% malware abused the DNS to achieve their evil intentions.

To mitigate these threats, tremendous efforts have been devoted in the last decades to establish domain reputation and blacklisting systems. The general approaches [5–8] extract multiple domain features (*e.g.*, TTL, lookup patterns,

H. Gao et al. (Eds.): CollaborateCom 2018, LNICST 268, pp. 488–503, 2019.
https://doi.org/10.1007/978-3-030-12981-1_34

number of IPs) from DNS records and then build a classifier over some labeled datasets. However, many of the features used are shown to be not robust [9] and can be easily altered by adversaries to evade the detection. Several recent techniques [10,11] propose to utilize graph methods to establish similarity between domains. These methods treat each domain as a vertex and add an edge between two domains if they exist an association (*e.g.*, hosted on a same IP [10]). Next, they use graph theories to calculate each domain's marginal probability distribution and infer its property (*i.e.*, malicious or benign). Comparing with previous works, graph-based methods exploit global associations of the DNS data to identify malicious domains, which are more robust. Unfortunately, graph-based methods cost a large amount of calculations, which making they not suitable for large scale passive DNS database (*e.g.*, millions of vertices).

In this paper, we propose `MalShoot`, a lightweight and robust technique for detecting malicious domains from passive DNS database. Our work is based on the fundamental intuition that domains sharing a similar resolution data (*e.g.*, hosted on some same IPs) are strongly associated and tend to have same property, namely malicious or benign. Inspired of the great success of embedding techniques in nature language processing, we treat a domain's DNS resolution data (rdata) as its context information and embed it into a low-dimensional vector. The domains that share similar context information are finally embedded into similar vector while. Based on the feature representation, a machine-learning classifier is trained over a labeled dataset and is further applied to detect other malicious domains. We evaluate `MalShoot` using real-world DNS traffic collected from three large ISP networks in China over two months under three popular machine learning algorithms (RandomForest, XGBoost and Deep Neural Network). `MalShoot` achieves 96.08% detection rate with an approximately 0.1% false positive rate with a 10-fold cross-validation when using RandomForest.

In summary, our paper makes the following contributions:

- We develop `MalShoot`, a lightweight and robust approach to detect malicious domains through graph embedding technique on passive DNS database.
- We design a novel domain representation technique that can automatically represent every domain into a feature vector while maintaining their DNS resolution information.
- We perform a comprehensive evaluation of `MalShoot` using two months real-world DNS traffic collected from three large ISP networks in China, demonstrating its effectiveness for detecting malicious domains.

We organize our paper as follows. Section 2 presents the background of passive DNS data and related works. Section 3 elaborates on the technical details of the proposed approach. Section 4 describes the datasets. Experiment setup and results analysis are reported in Sect. 5. We discuss a few issues of our approach in Sect. 6 and conclude the paper in Sect. 7.

2 Background and Related Work

2.1 Passive DNS Data

Passive DNS replication captures inter-server DNS messages through sensors that are voluntarily deployed by contributors in their DNS infrastructures. The captured DNS messages are further processed and then stored in a central passive DNS (PDNS) database which can be queried for various purposes [12].

```
## From ISP1
{“rrname”:”api.device.xiaomi.net”, “rrtype”:”A”,
“rdata”:”111.13.142.31”, ”count”:66
“time_first”:1481877769, ”time_last”:1485527818}
```

Fig. 1. A sample record collected from ISP_1

A typical record in a PDNS database is represented as a tuple: (*rrname, rrtype, rdata*, t_f, t_l, *count*), where the *rrname* is a domain name, *rrtype* represents the type of resource record (RR) returned by DNS servers, *rdata* is the data field in the RR, t_f and t_l denote the time when an individual *rdata* is first and last seen, and *count* is the number of DNS queries that receive the *rdata* in response. Figure 1 shows a sample record in our dataset. The *rdata* field is an IP address, which represents that the *rrname* ever was hosted on this IP.

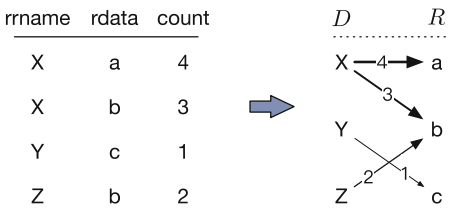


Fig. 2. Domain-resolution graph

Typically, we represent a passive DNS database as a domain-resolution bipartite graph $G = (D, R, E)$ by formatting each DNS record (*rrname, rrtype, rdata,* t_f, t_l, *count*) as an edge $e_{ij} = (rrname \rightarrow rdata)$ with weight $w_{ij} = count$, where D is the set of rrnames, R is the set of rdatas and E is the set of edges. Figure 2 shows an illustration for constructing a domain-resolution graph from DNS records.

2.2 Malicious Domain Detection

A wealth of research has been conducted on detecting malicious domains. Notos [5] is a pioneer work for establishing a dynamic reputation system to detect malicious domains. It use three categories of features to check a domain d, namely (i) network-based (*i.e.*, IPs associated with d); (ii) zone-based (*i.e.*, subdomains under d) and (iii) evidence-based (*i.e.*, malware samples contacting d). Bilge *et al.* proposed Exposure [6] with original time-based features, which requires less training time and data. On the other hand, Exposure overcomes some limitations of Notos, as it is able to identify malicious domains and addresses that were never seen in malicious activities before. Antonakakis *et al.* developed Kopis [7]) that aims at detecting malicious domains using the DNS traffic collected at the upper DNS hierarchy level (*i.e.*, Top Level DNS servers). The vantage point enables Kopis the *global* visibility. Moreover, Kopis can detect malware domains even when no IP reputation information is available. However, multiple features used in these detection systems, such as TTL and temporal patterns tend to be relatively brittle and allow attackers to take advantages of these features to evade detection.

Rahbarinia *et al.* proposed Segugio [8] for efficiently tracking the occurrence of new malware-control domain names in very large ISP networks. Their fundamental intuition is that infected machines tend to query new malware-control domains, moreover, machines infected with the same malware, or malware family, tend to query the same (or a partially overlapping) set of malware-control domains. On the other hand, benign machines have no reason to query malware-control domains. Manadhata *et al.* [13] proposed to identify malicious domains through host-domain query graph in an enterprise network. However, these efforts require the private information about individual users, which tends to be very sensitive.

Khalil *et al.* [10] and Peng *et al.* [11] proposed to discover malicious domains through the domain-resolution graphs. Compared to the host-domain graphs, domain-resolution graphs are constructed with publicly available DNS replication and without privacy concern. Khalil *et al.* [10] build domain-association graph by adding an edge between two domains if they hosted on same IPs for a period of time. The weight of the edge is decided on the number of IP they share. Then, they proposed a path-based inference to compute the global association with a set of malicious seeds for each unknown domain. Peng *et al.* [11] proposed a malicious domain detection method through DNS CNAME graph and focused on domains that are not resolved to IP addresses directly, but only appear in DNS CNAME records. The basic intuition is that domains connected by CNAME resource records share intrinsic relations and are likely to be in similar reputation. Unfortunately, these works are graphical analysis methods that cost complex calculation on the built domain-resolution graph when computing a domain's reputation score, which causes not scale well for large dataset. For example, [10] takes $O(|D|^2)$ steps to build the corresponding domain graph where $|D|$ is the number of domains. Our method first automatically learn the low-dimensional feature representation of every domain, which is linear with the

number of records (or, edges, in $O(|E|$ time). Afterwards, `MalShoot` calculates a domain's reputation only depending on the domain's low-dimensional representation, which is independent from the graph. Therefore, our method can scale well even with millions of domains.

2.3 Graph Embedding

Recent years have seen a surge of research on node embeddings on the graphs. Formally, these works aim to learn a mapping function that encodes each node on the graph to a low-dimensional vector. Early methods for learning representations for nodes largely focused on matrix-factorization approaches [14,15]. Many recent successful methods learn the node embeddings based on random walk statistics. DeepWalk [16] preserves higher-order proximity between nodes by maximizing the probability of observing the last k nodes and the next k nodes in the random walk centered at v_i, i.e. maximizing $\log Pr(v_{i-k}, \cdots, v_{i-1}, v_{i+1}, \cdots, v_{i+k})$. Node2vec [17] preserves higher-order proximity between nodes by maximizing the probability of occurrence of subsequent nodes in fixed length random walks. LINE [18] defines two functions, one each for first- and second-order proximities, and minimizes the combination of the two. `MalShoot` preserves the second-order proximity on the domain-resolution graph to embed every domain node into a low-dimensional feature vector.

3 Proposed Approach

In this section, we present our design of `MalShoot`. `MalShoot` is a lightweight method for identifying malicious domains using passive DNS database. It consists of three modules:

1. Representation Module: The representation module is designed for representing every individual domain name in PDNS database as a low-dimensional vector through graph embedding technique.
2. Training Module: The training module is responsible for training a malicious domain detection classifier using the learned vector representations over some labeled domains.
3. Classification Module: The classification module classifies remained unknown domains using the trained classifier.

Figure 3 provides an overview of `MalShoot`'s architecture, of required inputs, of outputs, and of the way `MalShoot` processes data internally. We describe the three modules in detail in the following.

3.1 Representation Module

A domain name consists of a set of strings separated by a period. Representation module is responsible for embedding every individual domain into a low-dimensional (*e.g.*, 128 dimensions) vector used for a downstream prediction task.

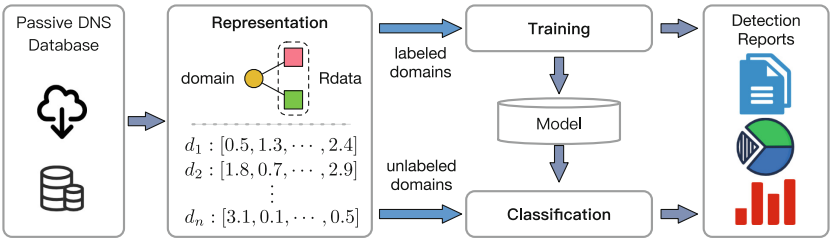


Fig. 3. Abstract illustration of the architecture of `MalShoot`

We formulate representation learning in our approach as a optimization problem. Let $G = (D, R, E)$ be the domain-resolution bipartite graph constructed from a given PDNS database, where $D = \{d_1, d_2, \cdots, d_n\}$ is the set of domains, $R = \{r_1, r_2, \cdots, r_m\}$ is the set of rdatas and E is the set of edges. Each edge, $(d_i, r_j) \in E$, represents a DNS record that domain d_i ever mapped to rdata r_j.

Let $f : D \rightarrow \mathbb{R}^d$ be the mapping function from domain nodes to d-dimensional vectors and $g : R \rightarrow \mathbb{R}^d$ be the mapping function from rdata nodes to d-dimensional vectors, where $d \ll |R|$ and $d \ll |D|$. Our goal is to map the domains that share a similar context information into similar vectors while the unrelated domains into distant vectors in the embedding space $\mathbb{R}^d$.

Denote $u_i = f(d_i)$ be the low-dimensional representation of domain d_i and $v_j = g(r_j)$ be the low-dimensional representation of rdata r_j. We first define the conditional probability of mapping to rdata r_j when giving domain d_i as:

$$p(r_j|d_i) = \frac{\exp(v_j^T \cdot u_i)}{\sum_{r_k \in R} \exp(v_k^T \cdot u_i)} \tag{1}$$

To preserve the context information in the embedding space, we should make the conditional distribution of the contexts $p(\cdot|d_i)$ specified by the low-dimensional representation be close to the empirical distribution $\hat{p}(\cdot|d_i)$. Therefore, we minimize the following objective function:

$$O = \sum_{d_i \in D} \lambda_i d(\hat{p}(\cdot|d_i), p(\cdot|d_i)), \tag{2}$$

where $d(\cdot, \cdot)$ is the distance between two distributions and λ_i indicates the importance or bias of domain d_i, which can be measured by occurrence times.

The empirical distribution $\hat{p}(\cdot|d_i)$ is defined as

$$\hat{p}(r_j|d_i) = \frac{w_{ij}}{W_i}, W_i = \sum_{r_k \in N(d_i)} w_{ik},$$

which w_{ij} is the weight of edge (d_i, r_j) and $N(d_i)$ is set of rdatas that a domain, d, ever mapped on the graph, G. In this paper, for simplicity we set $\lambda_i = W_i$

and we adopt KL-divergence as the distance function. Therefore, the objective function in Eq. (2) is as:

$$O = \sum_{d_i \in D} \lambda_i d(\hat{p}(\cdot|d_i), p(\cdot|d_i)) \tag{3}$$

$$= \sum_{d_i \in D} W_i \cdot - \sum_{r_j \in N(d_i)} \frac{w_{ij}}{W_i} \log \frac{p(r_j|d_i)}{\hat{p}(r_j|d_i)} \tag{4}$$

$$= \sum_{d_i \in D} \sum_{r_j \in N(d_i)} [-w_{ij} \log p(r_j|d_i) + w_{ij} \log \hat{p}(r_j|d_i)] \tag{5}$$

$$= \sum_{(d_i, r_j) \in E} [-w_{ij} \log p(r_j|d_i) + w_{ij} \log \hat{p}(r_j|d_i)] \tag{6}$$

Due to the w_{ij} and $\hat{p}(r_j|d_i)$ are constants after the domain-resolution graph is given, we omit these constants. Therefore, our objective function is to minimize:

$$O = - \sum_{(d_i, r_j) \in E} w_{ij} \log p(r_j|d_i) \tag{7}$$

By learning $\{u_i\}_{i=1\cdots|D|}$ and $\{v_j\}_{j=1\cdots|R|}$ that minimize this objective, we are able to represent every domain d_i with a d-dimensional vector u_i.

However, optimizing objective (7) is computationally expensive, which requires the summation over the entire rdata set of R when calculating the conditional probability $p(\cdot|d_i)$. To address this problem, we adopt the approach of negative sampling proposed in [19], which samples multiple negative edges according to some noisy distribution for each edge (d_i, r_j). More specifically, it specifies the following objective function for each edge (d_i, r_j):

$$w_{ij} \log \sigma(v_j^T \cdot u_i) + \sum_{k=1}^{K} E_{r_n \sim P_n(d_i)} \left[w_{in} \log \sigma(-v_n^T \cdot u_i) \right], \tag{8}$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the sigmoid function. The first term models the positive edges, the second term models the negative edges drawn from the noise distribution and K is the number of negative edges. We set $P_n(d_i) \approx W_i^{3/4}$ as proposed in [19]. We use sigmoid function in Eq. (8) is considered on that the derivation of sigmoid function can be easily computed, $\sigma(x)' = \sigma(x)(1 - \sigma(x))$. Therefore, minimizing objective function (8) is equal to minimize the $v_j^T \cdot u_i$ for positive edge (d_i, r_j) and maximize the $v_n^T \cdot u_i$ for negative edge (d_i, r_n).

We adopt the asynchronous stochastic gradient algorithm (ASGD) [20] for optimizing Eq. (8). In each step, the ASGD algorithm samples a mini-batch of edges and then updates the model parameters. Notice that, `MalShoot` can directly operate on the edges to learn the feature representations. In practical, we do not need to construct the domain resolution graph.

3.2 Training Module

As `MalShoot` is based on supervised learning classifiers, it requires training with labeled data. The training module implements training of classifiers and requires the input of benign domains and malicious domains. We obtain malicious domains from various sources, including Malware Domains List [21], Phishtank [22], Openphish [23]. To obtain an as clean as possible set of benign domains, we choose domains that are consistently ranked among the top 20 thousands in the world according to Alexa [24]. Section 4 describes how we collect benign and malicious domains.

The output of training module is a well-trained model, ready to be used for classification of unknown domains in the classification module.

3.3 Classification Module

The classification module classifies arbitrary domains into *benign* and *malicious* based on a model trained from the training module and pre-learned vector representation.

4 Dataset

4.1 Malicious Domains

We consistently collected malicious domains from multiple sources, including Malware Domains List [21], Phishtank [22], Openphish [23] everyday from Jan. 03, 2017 to Oct. 14, 2017. In addition, we also use the Zeus Block List [25] and the list of domains that are generated by the DGAs of Conficker [26]. These malicious domains lists represent a wide variety of malicious activity, including botnet command and control servers, drive-by download sites, phishing pages, scam sites that are found in spam mails and ransomware malware domains. Domains listed in Openphish and Phishtank are operated on URLs that are submitted by users. Hence, while most URLs in these repositories are malicious, not all of them are. We submit these domains to Google Safe Browsing [27] and only reserve the confirmed malicious ones.

4.2 Benign Domains

We collected legitimate domains according to Alexa [24]. We chose domains that are consistently ranked among the top 20 thousands from Jan. 16, 2015 to Mar. 5, 2017 (513 days). In addition, we manual filter out domains that allow for the "free registration" of subdomains, such as popular blog-publishing services or dynamic DNS domains (*e.g.*, wordpress.com and dyndns.com), as their subdomains are often abused by attackers. Finally, this produced a list of 9,216 popular domains.

4.3 Passive DNS Traffic

Thanks for CNCERT/CC [28], we accessed the passive DNS traffic of three large ISP networks in China. The three ISP networks are located in the provinces of Anhui, Guangdong and Shanghai respectively. We refer to these ISP networks simply as ISP_1, ISP_2 and ISP_3. Notice that this paper is part of an IRB-approved study; appropriate steps have been taken by our data provider to minimize privacy risks for the network users.

By inspecting the DNS traffic between the ISPs' customers and their local resolvers over two months (Dec 5, 2016 - Feb 5, 2017), we collected about 530 millions DNS queries, roughly 200 millions queries in ISP_1, 130 millions in ISP_2 and 200 millions in ISP_3. Due to the privacy concerns, we only preserved the data in network layer. We extracted the DNS RRs from the DNS response packets. Each RR is formated as $\{rrname, rrtype, rdata, ts\}$, the $rrname$ is the domain name, $rrtype$ and $rdata$ represent the type and data of the RR and the ts is the timestamp when observing this RR.

5 Experiment

In this section, we present the evaluation of `MalShoot` on the datasets collected from thee ISP networks. We first define the evaluation metrics and describe the experiments settings. Next, we compare the overall performance of three different classifiers on the three datasets. Then, we evaluate `MalShoot` on unknown datasets. Finally, we compare the performance to other two domain-resolution graph based works [10].

5.1 Evaluation Metrics

To quantify to performance of `MalHunter`, we define following three metrics:

- *True Positive Rate (TPR)*: the ratio of number of true positives of the total number of domains that identified as malicious. $TPR = \frac{N_{TP}}{N_{TP}+N_{FN}}$.
- *False Positive Rate (FPR)*: the ratio of number of true positives to the total number of domains that are actually malicious. $FRP = \frac{N_{FP}}{N_{FP}+N_{TN}}$.
- *Area Under the ROC Curve (AUC)*: the area under the receiver operating characteristic curve, which is compromise between TPR and FPR.

where N_{TP} is the number of malicious domains that are correctly identified as malicious, N_{FP} is the number of benign domains that are falsely identified as malicious, N_{FN} is the number of malicious domains that are falsely identified as benign and N_{TN} is the number of benign domains that are correctly identified as benign. Our goal is to achieve as high as TPR while maintaining low FPR.

5.2 Experiment Setting

We conducted experiments using the two-month DNS traffic collected from the three large ISP networks. For each dataset, we first constructed a domain-resolution graph and then applied the unsupervised feature learning module, described in Sect. 3.1, to represent every domain to a low-dimensional feature vector. We tried different choices of the dimension $d \in \{64, 128, 256\}$ and found the performances were very similar, therefore, we set $d = 128$ in the follow-up experiments. Based on the collected domain blacklists and whitelists, we obtained some known malicious and benign domains in our datasets, i.e., ground truth. Last, we performed standard 10-fold cross-validations based the labeled domains and their low-dimensional representations. Specifically, we partition the ground truth into ten fold, train the classifier using nine fold and test it on the remain one fold. We repeat the process for each fold and compute the average performance. For simplicity, we only focused on the `A`, `AAAA` and `CNAME` records and discard the other entries. Table 1 describes the detail statistics of the three domain-resolution graphs.

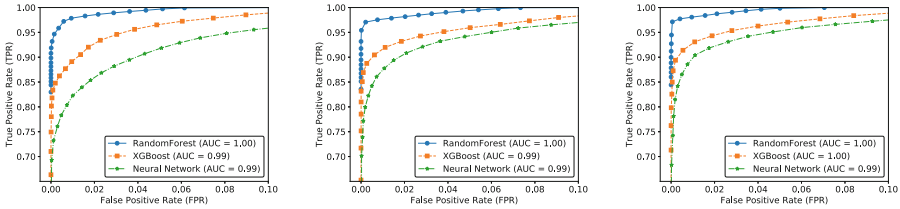
Table 1. Data description in the three ISPs. Each row in the table represents the ISP network of data source, the number of domains, rdatas, edges (records) in the domain-resolution graph, the known malicious and benign domains in the ISP.

ISP	Domains	RDatas	Edges	Malicious	Benign
ISP_1	2,001,117	411,720	2,518,641	4,674	3,651
ISP_2	1,542,334	559,338	1,767,760	4,587	4,659
ISP_3	1,841,241	486,946	2,485,603	3,538	5,383

We adopt three popular supervised machine learning algorithms: Random-Forest, XGBoost [29] and Deep Neural Network (DNN) as the classifiers to evaluate the performances.

1. RandomForest: We implement RandomForest using the scikit-learn [30] machine-learning library. The parameter settings are $n\_estimator = 200$, $min\_samples\_split = 11$ and $max\_features =$ 'sqrt'. The others are as default.
2. XGBoost: XGBoost [29] is an optimized distributed gradient boosting machine learning algorithms designed to be highly efficient, flexible and portable. We implement XGBoost with the open repository in Python. The parameter settings are $max\_depth = 6$ and $num\_boost\_round = 100$. The others are as default.
3. DNN: We implement a four layer perceptron model in PyTorch. Besides the input layer (receiving low-dimensional representations) and output layer (indicating the domain's reputation score), we add two hidden layers. The architecture is as follow: $Dense(D_{in}, nh_1) \rightarrow Relu(\cdot) \rightarrow Dense(nh_1, nh_2) \rightarrow Relu(\cdot) \rightarrow Dense(nh_2, 2) \rightarrow softmax(\cdot)$, where the $D_{in} = d = 128$ is

the length of features, $Dense(n, m)$ is a full connected layer with n input nodes and m output nodes, $Relu(\cdot)$ is a non-linear activation function, $Relu(x) = \max(x, 0)$ and $softmax(\cdot)$ is a normalized function to ensure the sum of the output to be 1. We empirically set $nh_1 = 128$ and $nh_2 = 32$. We optimize the model using ASGD [20] with $batch\_size = 32$ and $cross\_entropy$ loss for 100 epochs.



(a) ISP_1 (b) ISP_2 (c) ISP_3

Fig. 4. Performance comparison of classifiers under 10-fold cross-validation.

5.3 Experimental Results

We test the effectiveness of our detector over the ground-truth datasets through the standard 10-fold cross-validation. We compare three mostly used machine-learning classification algorithms, including RandomForest, XGBoost and Neutral Network. Figure 4 illustrates the receiver operating characteristic (ROC) curves of these classifiers, when using three PDNS database collected from three ISPs. The x-axis shows the false positive rate (FPR) and the y-axis shows the true-positive rate (TPR). We observe that all classifiers can achieve promising accuracy on the three data sources. To reach a 90% detection rate, the maximum FRP is always less than 4% for all classifiers, suggesting that `MalShoot` can effectively detect malicious domains. RandomForest outperforms the other classifiers in all cases. We achieves a 96.08% true positive rate with an approximately 0.1% false positive rate when using RandomForest.

5.4 Evaluation on Unknown Dataset

We now evaluate `MalShoot` on unknown dataset to examine whether we can accurately detect other unknown malicious domains based on the trained classifier. We focus on the best performing classifier RandomForest only and use it to for all follow-up experiments. We first use the full labeled dataset to train our detection model and then apply it to unknown domains. Due to the space limit, we only present the results on dataset of ISP_1 (results on other two ISPs have a similar distribution) in the rest of evaluations.

Among the 1,992,792 unknown domains in ISP_1 dataset (Table 1), `MalShoot` reports 173,279 malicious domains. Since this dataset is unlabeled, we have to validate the result through manual investigation. We use following two rules to consider a domain as a true positive (1) if it is hosted on a black IP address, and (2) if its second level domain (SLD) is reported as malicious (*e.g.*, if `abc.com` is reported as malicious, we treat its all subdomains as malicious). Notice that, we are conservative to judge an IP address as a black IP. We treat an IP as black only if it ever hosted enough malicious domains, (*e.g.*, 10 in our rule), which can reduce the false positives. Rule (1) confirms 168,002 as true positives and rule (2) confirms 1,219 of the left 5,277 as true positives. For the remain 4,058 not matched domains, we randomly select 100 domains and manually validate them. We find 34 domains host web content and most of them can be classified under gambling and sex categories. Therefore, `MalShoot` can effectively discover newly malicious domains with very low false positive rate.

5.5 Comparison

We compare `MalShoot` to one previous domain-resolution graph method [10].

1. Khalil *et al.* [10]:
   (a) First phase: constructing domain association graphs from DNS A records by adding an edge between two domains if they share common IPs.
   (b) Second phase: computing the global association scores with a given set of malicious seeds for every unknown domain through path-based inference.
2. `MalShoot`:
   (a) First phase: embedding every domain into a low-dimensional feature vector while maintaining its context information.
   (b) Second phase: training a classifier over a labeled dataset based on the feature representations and then applying it to detect other unknowns.

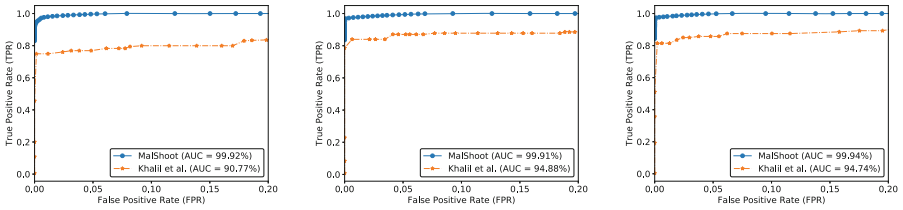
We first compare the time complexity of these two methods. For a regular domain-resolution graph $G = (D, R, E)$, the time complexity is listed in Table 2. Due to the complexity of training classifier relies on the machine-learning algorithm, we only analyze it for RandomForest.

In the first phase, approach in [10] computes the weight of every two domains if they share common IPs, therefore it costs about $O(|D|^2)$ steps. `MalShoot` updates model parameters edge by edge using Eq. (8), which costs $O(|E|)$ steps. Notice that $|D| \approx |E|$ in domain-resolution graphs (detail numbers are listed in Table 1). In the second phase, `MalShoot` works independently from the domain-resolution graph while method in [10] still needs to run complex calculation (*e.g.*, find shortest path) on the graph. Therefore, `MalShoot` can detect malicious domains without constructing domain-resolution graph or complex graphical calculation, which endows it scales well in large dataset.

We implement the method [10] and run it on the three datasets. Figure 5 shows the performances of comparing with our method. We observe that `MalShoot` outperforms the Khalil *et al.* [10] in the three datasets. The weight

Table 2. Time complexity analysis of `MalShoot` and [10]. Where $|D|$, $|E|$ is the number of domains and edges in the domain resolution graph. M is the number of trees used in RandomForest and N is the number of training samples. $|S|$ is the number of malicious seeds and $|E'|$ is the number of edges in domain graph, which could be as large as $|D|^2$

Method	First phase	Second phase
`MalShoot`	$O(\|E\|)$	Training: $O(MN \log N)$ Predicting: $O(M \log N)$
Khalil *et al.* [10]	$O(\|D\|^2)$	$O(\|S\|(\|E'\| + \|D\| \log \|D\|))$



(a) ISP_1 (b) ISP_2 (c) ISP_3

Fig. 5. Performance comparison to [10] and [11]

between the two domains in [10] is defined as $w(d_i, d_j) = 1 - \frac{1}{1+|asn(ip(d_i) \cap ip(d_j))|}$, where $ip(d_i)$ is the set of IPs that domain d_i is resolved to and $asn(I)$ is the set of ASNs that the IPs in I belong to. Notice that each domain-IP pair is treated with same weight, neglecting the preferences. For example, domain d may be typically hosted on ip_1 while temporarily on ip_2, however, (d, ip_1) and (d, ip_2) are treated the same in [10], which causes it not robust for noisy resolutions (*e.g.*, fake resolution information). `MalShoot` takes a fine-grained consideration on the weight (*e.g.*, w_{ij}) of different domain-rdata pairs, which makes it robust for noisy resolutions.

6 Discussion and Future Work

In this section, we discuss some practical issues of our method.

First, `MalShoot` is a graph embedding method to detect malicious domains. It embeds every domain into a low-dimensional vector based on its context information. One practical issue is how to accurately embed domains with limited context information. For example, it is very hard to accurately infer a domain's representation if it never shares rdatas with others or only maps to a few rdatas. Notice that `MalShoot` only extract the *second-order* proximity in the domain-resolution graph. An intuitive solution to this is also preserving the *first-order* proximity, *i.e.*, maintaining the association between domain and rdata, when embedding. Therefore, if a domain maps to IPs listed in blacklists, we can also detect it.

Second, `MalShoot` is trained based on historical DNS data. Therefore, another practical issue is how to find the representation of a new domain. For a new domain d_i, if it maps to the existing rdatas, we can obtain the empirical distribution $\hat{p}(\cdot|d_i)$ over existing rdatas. To obtain the embedding of the new domain, according to the objective function Eq. (7), a straightforward way is to minimize the following objective function

$$\sum_{r_j \in N(d_i)} w_{ij} \log p(r_j|d_i)$$

by updating the embedding of the new domain and keeping the embeddings of existing rdatas. If the rdata that the new domain maps is also new, we must resort to other information, such as the lookup pattern behavior of the domain and we leave it as our future work.

7 Conclusion

In this paper, we propose a graph embedding based technique to discover malicious domains through public DNS records. We treat the set of rdatas that a domain ever mapped as its context and assume that domains sharing a similar context information are strongly associated. We design a unsupervised feature learning module that automatically represent every individual domain into a low-dimensional vector while maintaining their context information. Based on the learned features, a classifier is built over a labeled dataset and further be applied for detecting other unknown malicious domains. Experimental results show that our technique can achieve high true positive rates with low false positive rates. Compared to previous works, our approach scales well in large datasets.

Acknowledgments. The research leading to these results has received funding from the National Key Research and Development Program of China (No. 2016YFB0801502) and National Natural Science Foundation of China (No. U1736218). We thank the CNCERT/CC for providing the DNS data used in our experiments. The corresponding author is Xiaochun Yun.

References

1. Grier, C., Thomas, K., Paxson, V., Zhang, M.: @ spam: the underground on 140 characters or less. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, pp. 27–37. ACM (2010)
2. Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A comprehensive measurement study of domain generating malware. In: USENIX Security Symposium, pp. 263–278 (2016)
3. Antonakakis, M., et al.: From throw-away traffic to bots: detecting the rise of DGA-based malware. In: Presented as Part of the 21st USENIX Security Symposium (USENIX Security 2012), pp. 491–506 (2012)
4. Cisco 2016 annual security (2016). http://www.cisco.com/c/m/en_us/offers/sc04/2016-annual-security-report/index.html

5. Antonakakis, M., Perdisci, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation system for DNS. In: USENIX Security Symposium, pp. 273–290 (2010)
6. Bilge, L., Sen, S., Balzarotti, D., Kirda, E., Kruegel, C.: Exposure: a passive DNS analysis service to detect and report malicious domains. ACM Trans. Inf. Syst. Secur. (TISSEC) **16**(4), 14 (2014)
7. Antonakakis, M., Perdisci, R., Lee, W., Vasiloglou II, N., Dagon, D.: Detecting malware domains at the upper DNS hierarchy. In: USENIX Security Symposium, vol. 11, pp. 1–16 (2011)
8. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: efficient behavior-based tracking of malware-control domains in large ISP networks. In: 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, pp. 403–414. IEEE (2015)
9. Stinson, E., Mitchell, J.C.: Towards systematic evaluation of the evadability of bot/botnet detection methods. WOOT **8**, 1–9 (2008)
10. Khalil, I., Yu, T., Guan, B.: Discovering malicious domains through passive DNS data graph analysis. In: Proceedings of the 11th ACM on Asia Conference on Computer and Communications Security, pp. 663–674. ACM (2016)
11. Peng, C., Yun, X., Zhang, Y., Li, S., Xiao, J.: Discovering malicious domains through alias-canonical graph. In: 2017 IEEE Trustcom/BigDataSE/ICESS, pp. 225–232. IEEE (2017)
12. Weimer, F.: Passive DNS replication. In: FIRST Conference on Computer Security Incident, p. 98 (2005)
13. Manadhata, P.K., Yadav, S., Rao, P., Horne, W.: Detecting malicious domains via graph inference. In: Kutyłowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 1–18. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11203-9_1
14. Cao, S., Lu, W., Xu, Q.: Grarep: learning graph representations with global structural information. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, pp. 891–900. ACM (2015)
15. Ou, M., Cui, P., Pei, J., Zhang, Z., Zhu, W.: Asymmetric transitivity preserving graph embedding. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1105–1114. ACM (2016)
16. Perozzi, B., Al-Rfou, R., Skiena, S.: DeepWalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)
17. Grover, A., Leskovec, J.: Node2vec: scalable feature learning for networks. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 855–864. ACM (2016)
18. Tang, J., Qu, M., Wang, M., Zhang, M., Yan, J., Mei, Q.: Line: large-scale information network embedding. In: Proceedings of the 24th International Conference on World Wide Web, pp. 1067–1077. International World Wide Web Conferences Steering Committee (2015)
19. Mikolov, T., Sutskever, I., Chen, K., Corrado, G.S., Dean, J.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
20. Recht, B., Re, C., Wright, S., Niu, F.: Hogwild: a lock-free approach to parallelizing stochastic gradient descent. In: Advances in Neural Information Processing Systems, pp. 693–701 (2011)
21. Malware domain block list (2018). http://www.malwaredomains.com

22. Phishtank (2018). http://www.phishtank.com
23. Openphish (2018). https://openphish.com
24. Alexa top 1 million (2017). http://s3.amazonaws.com/alexa-static/top-1m.csv.zip
25. Zeus domain blocklist (2016). https://zeustracker.abuse.ch/blocklist.php
26. Porras, P.A., Saïdi, H., Yegneswaran, V.: A foray into Conficker's logic and rendezvous points. In: LEET (2009)
27. Google safe browsing (2018). https://www.google.com/transparencyreport/safebrowsing/diagnostic/
28. CNCERT/CC (2018). https://www.cert.org.cn
29. Chen, T., Guestrin, C.: XGBoost: a scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785–794. ACM (2016)
30. Pedregosa, F., et al.: Scikit-learn: machine learning in Python. J. Mach. Learn. Res. **12**, 2825–2830 (2011)