HAL
open science

A Topology Based Flow Model for Computing Domain Reputation

Igor Mishsky, Nurit Gal-Oz, Ehud Gudes

To cite this version:

Igor Mishsky, Nurit Gal-Oz, Ehud Gudes. A Topology Based Flow Model for Computing Domain Reputation. 29th IFIP Annual Conference on Data and Applications Security and Privacy (DBSEC), Jul 2015, Fairfax, VA, United States. pp.277-292, 10.1007/978-3-319-20810-7_20 . hal-01745814

HAL Id: hal-01745814

https://inria.hal.science/hal-01745814v1

Submitted on 28 Mar 2018

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

A Topology based Flow Model for Computing Domain Reputation

Igor Mishsky[1], Nurit Gal-Oz[2] and Ehud Gudes[1]

[1] Ben-Gurion University, Beer-Sheva, 84105, Israel, igormishsky@gmail.com, ehud@cs.bgu.ac.il
[2] Sapir Academic College, D.N. Hof Ashkelon 79165, Israel galoz@sapir.ac.il

Abstract. The Domain Name System (DNS) is an essential component of the internet infrastructure that translates domain names into IP addresses. Recent incidents verify the enormous damage of malicious activities utilizing DNS such as bots that use DNS to locate their command & control servers. Detecting malicious domains using the DNS network is therefore a key challenge.
We project the famous expression *Tell me who your friends are and I will tell you who you are*, motivating many social trust models, on the internet domains world. A domain that is related to malicious domains is more likely to be malicious as well.
In this paper, our goal is to assign *reputation* values to domains and IPs indicating the extent to which we consider them malicious. We start with a list of domains known to be malicious or benign and assign them reputation scores accordingly. We then construct a DNS based graph in which nodes represent domains and IPs.
Our new approach for computing domain reputation applies a flow algorithm on the DNS graph to obtain the reputation of domains and identify potentially malicious ones. The experimental evaluation of the flow algorithm demonstrates its success in predicting malicious domains.

1 Introduction

Malicious botnets and Advanced Persistent Threats (APT) have plagued the Internet in recent years. Advanced Persistent Threat, often implemented as a botnet, is advanced since it uses sophisticated techniques to exploit vulnerabilities in systems, and is persistent since it uses an external command and control (C&C) site which is continuously monitoring and extracting data of a specific target. APTs are generated by hackers but are operated from specific domains or IPs. The detection of these misbehaving domains (including zero day attacks) is difficult since there is no time to collect and analyze traffic data in real-time, thus their identification ahead of time is very important. We use the term *domain reputation* to express a measure of our belief that a domain is benign or malicious. The term reputation is adopted from the field of social networks and virtual communities, in which the reputation of a peer is derived from evidences regarding its past behavior but also from its relations to other peers [12].

A domain reputation system can support the decision to block traffic or warn organizations about suspicious domains. Currently lists of domains which are considered legitimate are published by web information companies (e.g., Alexa [1]) while black-lists of malware domains are published by web threat analysis services (e.g., VirusTotal [14].) Unfortunately the number of domains appearing in both types of lists is relatively small, and a huge number of domains is left unlabeled. Therefore, the problem of assigning reputation to unlabeled domains is highly important.

The Domain Name Service (DNS) maps domain names to IP addresses and provides an essential service to applications on the internet. Many botnets use a DNS service to locate their next C&C site. For example, botnets tend to use short-lived domains to evasively move their C&C sites. Therefore, DNS logs have been used by several researchers to detect suspicious domains and filter their traffic if necessary. Choi and Lee [4], analyzed DNS traffic to detect APTs. Such analysis requires large quantities of illegitimate DNS traffic data.

An alternative approach was proposed in the Notos system [2] which uses historical DNS information collected passively from multiple recursive DNS resolvers to build a model of how network resources are allocated and operated for legitimate Internet services. This model is mainly based on statistical features of domains and IPs that are used for building a classifier which assigns a reputation score to unlabeled domains. The main difference between the DNS data used for computing reputation and the data used for malware detection is that the first consists of mainly static properties of the domain and DNS topology data, while the latter requires behavioral and time-dependent data. While DNS behavior data may involve private information (e.g., the domains that an IP is trying to access), which ISPs may be reluctant to analyze or share, DNS topology data is much easier to collect. Our research also focuses on computing domain reputation using DNS topology data.

Various definitions of the terms trust and reputation have been proposed in the literature as the motivation for a computational metric. Trust is commonly defined following [11] as *a subjective expectation an agent has about another's future behavior based on the history of their encounters.* The history is usually learned from ratings that peers provide for each other. If such direct history is not available, one derives trust based on reputation. Reputation is defined [11] as *the aggregated perception that an agent creates through past actions about its intentions and norms.* , where this perception is based on information gathered from trusted peers. These definitions are widely used in state of the art research on trust and reputation as the logic behind trust based reputation computational models in web communities and social networks. However, computing reputation for domains raises several new difficulties:

- Rating information if exists, is sparse and usually binary, a domain is labeled either "white" or "black"
- Static sources like blacklists and whitelists are often not up-to-date
- There is no explicit concept of trust between domains which make it difficult to apply a flow or a transitive trust algorithm.

– Reputation of domains is dynamic and changes very fast

These difficulties make the selection of an adequate computational model for computing domain reputation a challenging task. The focus of our paper and its main contribution is therefore a flow model and a flow algorithm for computing domain reputation which uses a topology-based network that maps connections of domains to IPs and other domains. Our model uses DNS IP-Domain mappings and statistical information but does not use DNS traffic data.

Our approach is based on a flow algorithm, commonly used for computing trust in social networks and virtual communities. We are mainly inspired by two models: the Eigentrust model [8] that computes trust and reputation by transitive iteration through chains of trusting users; and the model by Guha et al. [9] which combines the flow of trust and distrust. The motivation for using a flow algorithm is the hypothesis that IPs and domains which are neighbors of malware generating IPs and domains, are more likely to become malware generating as well. We construct a graph which reflects the topology of domains and IPs and their relationships and use a flow model to propagate the knowledge received in the form of black list, to label domains in the graph as suspected domains. Our preliminary experimental results support our proposed hypothesis that domains (or IPs) connected to malicious domains have a higher probability to become malicious as well.

The main contribution of this paper lies in the novelty of the algorithm and the strength of the supporting experimental study.

The experimental study supporting our results, uses a sizable DNS database (more than a one million IPs and domains) which proves the feasibility of our approach. The rest of this paper is organized as follows. Section 2 provides a more detailed background on DNS and IP characteristics and on the classical flow models, and then surveys the related work. Section 3 discusses the graph construction, the weight assignment problem and the flow algorithm. Section 4 presents the results of our experimental evaluation and Section 5 concludes the paper and outlines future research.

2 Background and Related Work

The domain name system (DNS) translates Internet domains and host names into IP addresses. It is implemented as an hierarchical and distributed database containing various types of data, including host names and domain names, and provides application level protocol between clients and servers. An often-used analogy to explain the Domain Name System is that it serves as the phone book for the Internet by translating human-friendly computer host names into IP addresses. Unlike a phone book, the DNS can be quickly updated, allowing a service's location on the network to change without affecting the end users, who continue to use the same host name. Users take advantage of this when they use meaningful Uniform Resource Locators (URLs). In order to get the IP of a domain, the host usually consults a local recursive DNS server (RDNS).The RDNS iteratively discovers which authoritative name server is responsible for

each zone. The result of this process is the mapping from the requested domain to the IP requested.

Two categories of models are related to our work. The first category deals with ways to compute domain reputation. The second deals with flow algorithms for the computation of trust and reputation in general. Domain reputation is a relatively new research area. The Notos model for assigning reputation to domains [2] was the first to use statistical features in the DNS topology data and to apply machine learning methods to construct a reputation prediction classifier. Notos uses historical DNS information collected passively from multiple DNS resolvers to build a model of how network resources are allocated and operated for professionally run Internet services. Specifically it constructs a set of clusters representing various types of popular domains statistics and computes features which represent the distance of a specific domain from these clusters. Notos also uses information about malicious domains obtained from sources such as spam-traps, honeynets, and malware analysis services to build a set of features representing how network resources are typically allocated by Internet miscreants. With the combination of these features, Notos constructs a classifier and assigns reputation scores to new, previously unseen domain names.(Note that Notos uses heavily, information about highly popular sites such as Acamai which is not publicly available and therefore make it difficult to compare to.) The Exposure system [10] collects data from the DNS answers returned from authoritative DNS servers and uses a set of 15 features that are divided into four feature types: time-based features, DNS answer-based features, TTL value-based features, and domain name-based features. The above features are used to construct a classifier based on the J48 decision tree algorithm [16] in order to determine whether a domain name is malicious or not. Kopis [3] is a system for monitoring the high levels of the DNS hierarchy in order to discover the anomaly in malicious DNS activities. Unlike other detection systems such as Notos [2] or Exposure [10],Kopis takes advantage of the global visibility of DNS traffic at the upper levels of the DNS hierarchy to detect malicious domains. After the features are collected it uses the random forest technique as the machine learning algorithm to build the reputation prediction classifier.

In the category of flow algorithms for computation of trust in general, two models are of specific interest to our work. The first is Eigentrust [8], a reputation management algorithm for peer-to-peer network. The algorithm provides each peer in the network a unique global trust value based on the peer's history of uploads and thus aims to reduce the number of inauthentic files in a P2P network. The algorithm computes trust and reputation by transitive iteration through chains of trusting users. The page-rank algorithm [12] uses a similar approach, however it contains special features related to URL referencing. Guha et al. [9] introduce algorithms for implementing a web-of-trust that allows people to express either trust or distrust in other people. Two matrices representing the trust and distrust between people are built using four types of trust relationships. They present several schemes for explicitly modeling and propagating trust and distrust and propose methods for combining the two, using weighted

linear combination. The propagation of trust was also used by Coskun et al. [6] for detecting potential members of botnets in P2P networks. Their proposed technique is based on the observation that peers of a P2P botnet with an unstructured topology, communicate with other peers in order to receive commands and updates. Since there is a significant probability that a pair of bots within a network have a mutual contact, they construct a mutual contact graph. This graph is different than the DNS topology graph we rely on, the attributes and semantics underlying our approach are different and accordingly the algorithm we propose. Wu et al. [17] use the distrust algorithm presented by Guha et al. [9] for detecting spam domains but use URL references rather than DNS data to derive the edges between domain nodes. They also discuss trust attenuation and the division of trust between a parent and its "children". Yadav et al. [18] describe an approach to detect malicious domains based mainly on their names distribution and similarity. They claim that many botnets use the technique of DGA (domain generating algorithm) and they show that domains generated in this form have certain characteristics which help in their detection. The domain names usually have a part in common, e.g. the top level domain (TLD), or a similar distribution of alpha-numeric characters in their names. The success of using the above characteristics in [18] motivates the construction of domain-domain edges in our graph as well.

There are quite a few papers which use DNS data logs to detect Botnets and malicious domains. However these papers use the DNS traffic behavior and not the mapping information used by Notos and in our work. Villarmin et al. [13] provide C&C detection technique motivated by the fact that bots typically initiate contact with C&C servers to poll for instructions. As an example, for each domain, they aggregate the number of non-existent domains (NXDOMAIN) responses per hour and use it as one of the classification features. Another work of this category, presented by Choi and Lee [4] monitor DNS traffic to detect botnets, which form a group activity in similar DNS queries simultaneously. They assume that infected hosts perform DNS queries at several occasions and using this data they construct a feature vector and apply a clustering technique to identify malicious domains. As discussed above, although DNS traffic data has significant features, it is difficult to obtain comparing to DNS topological data.

To summarize, although there are some previous works on domain reputation using DNS statistical features (e.g., Notos), and there exist flow algorithms in other trust and reputation domains, the combination of the two as used in this paper is new.

3 The Flow Model

The goal of the flow algorithm is to assign domains with reputation scores given an initial list of domains with known reputation (good or bad). The first step is the construction of the Domain-IP graph based on information obtained from a large set of successful DNS transactions represented as *A-records*. The A-records are used to construct the DNS topology graph, where vertices represent IPs and

domains, and the weighted edges represent the strength of their connections. This is described next. In subsection 3.2 we present in detail the flow algorithm, and describe the method for combining good and bad reputation scores. Finally we discuss an optimization of the algorithm needed for large graphs.

3.1 Constructing the graph and assigning edge weights

The DNS topology graph consists of two types of vertices: domains and IPs, deriving four types of edges between them. To construct the graph we use A-records, and also data available from public sources to estimate the strength of connections between any two vertices, IP or domain, by the amount of common data between them. The **IP data** for each IP consists of the following five characteristics available from sources such as e.g., WHOIS databae [15]:

- Autonomous System number (ASN): a collection of connected Internet Protocol (IP) routing prefixes under the control of one or more network operators that present a common, clearly defined routing policy to the Internet. The ASN number is the indexation of the collection.
- Border Gateway Protocol (BGP) prefix: a standardized exterior gateway protocol designed to exchange routing and reachability information between autonomous systems (AS) on the Internet. The protocol defines the routing of of each ASN.BGP prefix as a range of IPs to which it routes.
- Registrar: the name of the organization that registered the IP.
- Country: the country to which the IP belongs.
- Registration date: the date and time at which the IP was registered.

For **Domain data** the key concept is the parent of a domain. A k-Top Level Domain (kTLD) is the k suffix of the domain name [2]. For example: for domain finance.msn.com, the 3TLD is the same as the domain name: finance.msn.com, the 2TLD is .msn.com and the 1TLD is .com.

We use the following notation:
Set_{IP} is the set containing all the IPs.
Set_{domain} is the set containing all the domains.
$Set_{parent} \subseteq Set_{domain}$ is the set containing all the parents.
$Set_{commonAtt}$ is the set of attributes vectors derived from IP data. Attribute appear in the following order: country, ASN, BGP prefix, registrar, registration date. Missing information is replaced with 'none'. For example
$(DE, none, none, STRATO.DE, none) \in Set_{CommonAtt}$ is the vector element in which the only information available is the country and the registrar.

We define a weight function that assigns a weight to each edge in the graph. Let w be a weight function $w : (u, v) \rightarrow [0, 1]$ used to assign weight to the edge (u, v) where $u, v \in Set_{IP} \cup Set_{domain}$, for each edge type we consider three alternative weight functions as follows:

1. IP to domain: For $ip \in Set_{IP}$ and the list of A-records, let D_{ip} be all the domains mapped to ip. For each $d \in D_{ip}$ we define: $w(ip, d) = \frac{1}{|D_{ip}|}; \frac{1}{\log |D_{ip}|}; 1.$

2. Domain to IP: For $d \in Set_{domain}$ and a list of A-records, let I_d be all the IPs that were mapped to d. For each $ip \in I_d$ we define: $w(d, ip) = \frac{1}{|I_d|}; \frac{1}{\log |I_d|}; 1$.

3. IP to IP: Let *commonAtt* be a combination of the five attributes of IP data. Let $Set_{commonAtt}$ be the set of all IPs with the attribute combination $Common_Att$. For each $ip_1, ip_2 \in Set_{commonAtt}$ s.t. $ip_1 \neq ip_2$ we define: $w(ip_1, ip_2) = \frac{1}{|Set_{commonAtt}|}; \frac{1}{\log |Set_{commonAtt}|}; 1$.

4. Domain to Domain: Let P_d be the set of all domains with the same parent domain d. For each $d1, d2 \in P_d$ s.t. $d1 \neq d2$ we define: $w(d_1, d_2) = \frac{1}{|P_d|}; \frac{1}{\log |P_d|}; 1$.

The intuition behind the above definition of weights is that, the effect of a domain reputation on the IPs it is mapped to, increases, as the amount of mapped IPs decreases. We use three approaches for computing weight, which produce 81 different combinations from which a subset was tested. We represent our graph as an adjacency matrix M, in which the value of an entry (i, j) is the weight between vertex i and j computed according to the selected combination.

3.2 The Flow algorithm

The flow algorithm models the idea that IPs and domains affect the reputation of IPs and domains connected to them. This is done by propagating a node's reputation iteratively, so that the reputation in each iteration is added to the total reputation accumulated within a domain or IP node, using some attenuation factor. The attenuation factor is a means to reduce the amount of reputation accumulated by transitivity. The flow algorithm is executed separately to propagate good reputation and bad reputation. The algorithm is presented in two parts, the first is the *Basic Flow*, which describes the flow algorithm in general, and the second is the *Combined Algorithm*, which focuses on the way bad reputation and good reputation are combined.

Figure 1 outlines the preparation steps prior to the execution of the basic algorithm. The **Basic Flow** algorithm starts with an initial set of domains which are labeled either bad or good. The parameters of the algorithm are:

1. A matrix M, where each entry represents a weighted edge between the vertices (domains or IPs); M^T denotes the transpose of the matrix M.
2. $V_{initial}$ - a vector representing the initial reputation value of each vertex, based on the initial set of labeled vertices.
3. $n \in \mathcal{N}$ - the number of iterations.
4. $atten \in [0, 1]$ - the attenuation factor.
5. $\theta \in [0, 1]$ - a reputation threshold above which a vertex is considered bad.

Algorithm 1 outlines the basic flow algorithm that propagates the reputation from a node to its neighbors and is carried out in three steps:

1. Calculate the matrix for the ith iteration:
   $M_i = (M^T \cdot atten)^i$

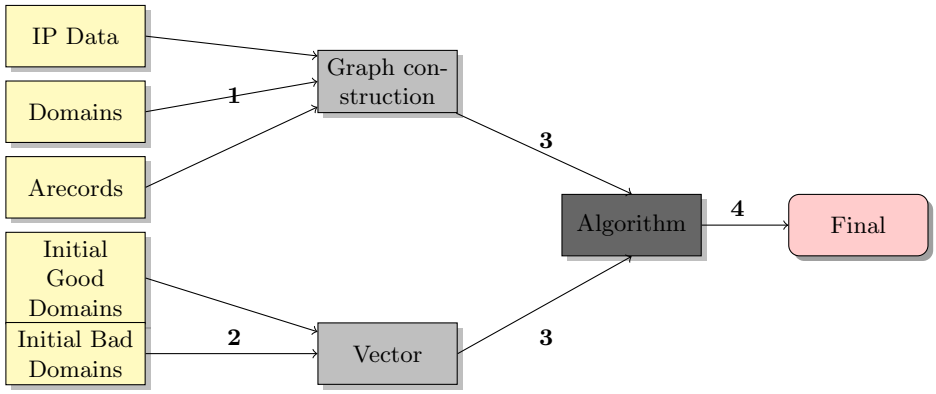


Fig. 1: The process for computing reputation scores: (1) Create the topology graph, assign weights and represent as a matrix; (2) Create the initial vector used for propagation; (3) Use the vector and the matrix as input to the flow algorithm; (4) output final reputation scores.

2. Calculate the final Matrix after n iterations as the sum of all the matrices with attenuation:
   $M_{final} = \sum_{i=0}^{n} (M^T \cdot atten)^i$
3. Calculate the final reputation vector: $V_{final} = M_n \cdot V_{initial}$, the reputation scores of all vertices after the propagation.

Applying the algorithm separately to propagate bad and good reputation may result in a node that is labeled both as good and bad. The final labeling of such node depends on the relative importance given to each label as done in the combined algorithm.

Algorithm 1 Basic

```
procedure BASIC(M, n, atten, V_initial)
    for i = 0 to n do
        M_i ← (atten · M^T)^i
    M_final ← Σ_{i=0}^{n} M_i
    V_final ← M_final · V_initial
    return V_final
```

The **combined algorithm** runs the basic flow algorithm twice with $V_{initial} = V_{good}$ and $V_{initial} = V_{bad}$. Each flow is configured independently with the following parameters: factor, threshold, and number of iterations. We denote $n_{good}, n_{bad} \in \mathcal{N}$ as the number of iterations for the good and bad flow respectively; $atten_{good}$, $atten_{bad} \in \mathcal{R}$, as the attenuation factor for the good and bad flow respectively; and $w \in \mathcal{R}$ the weight of the "good" reputation used when combining the results. Algorithm 2 uses of the basic flow algorithm to compute the good and

bad reputation of each vertex and merge the results. Set_{Mal} is the result set of domains identified as bad.

Algorithm 2 Combined

1: $V_{good} \leftarrow basic(M, n_{good}, atten_{good}, V_{good})$
2: $V_{bad} \leftarrow basic(M, n_{bad}, atten_{bad}, V_{bad})$
3: $Set_{Mal} \leftarrow \emptyset$
4: **for** $d \in Domains$ **do**
5: **if** $V_{bad}[d] + w \cdot V_{good}[d] > \theta$ **then** $Set_{Mal} \leftarrow Set_{Mal} \cup \{d\}$
6: return Set_{Mal}

The combined algorithms ignores the following observations which lead us to examine another approach:

- A reputation score is a value in the range of [0,1], therefore a vertex should not distribute a reputation score higher then 1 to it's neighbors.
- Initial labels are facts and therefore domains that were initially labeled as good or bad should maintain this label throughout the propagation process.
- A domain gaining a reputation value above a predefined threshold, bad or good, is labeled accordingly and maintain this label throughout the propagation process.

The extended algorithm shown as Algorithm 3 is proposed to address these observations. The main new procedure is the *Normalize* procedure which normalizes the scores after each iteration. The threshold used is the average of the scores so far (good or bad according to the actual procedure).

3.3 Optimization for large graphs

As we deal with millions of domains and IPs we have to calculate the scores in an efficient way. The most computationally intensive step is the matrix multiplication (see Figure 1). To speed it up we use a special property of our graph which is the existence of *Cliques*. There are two kinds of cliques in the graph: cliques of IPs which share the same set of common attributes and cliques of domains which share the same parent or the same name server. Since a clique can contain thousands of nodes we calculate the flow within it separately and not as part of the matrix multiplication. A *clique* in a graph is a subset of its vertices such that every two vertices in the subset are connected by an edge.
We define a **Balanced Clique** as a clique such that all vertices have the same attribute values. This necessarily leads to a clique with a single weight value on all of its edges.

Theorem 1. *Let M be a matrix representing a weighted directed graph and V a vector representing vertices values, and let BC be the set of vertices that form a balanced clique, such that the weight on every edge in BC is $const_{BC} \in R$; For*

Algorithm 3 Extended

```
for x ∈ Domains do
    V_bad[x] ← 0; V_good[x] ← 0
    if x ∈ Set_bad then V_bad[x] ← 1
    if x ∈ Set_good then V_good[x] ← 1
for i = 1 to n do
    V_good ← V_good + (M^T) · V_good
    V_bad ← V_bad + (M^T) · V_bad
    Normalize(Set_good, V_bad, V_good)
    Normalize(Set_bad, V_good, V_bad)
Set_Mal ← ∅
for d ∈ Domains do
    if V_bad[d] + w · V_good[d] > θ then Set_Mal ← Set_Mal ∪ {d}
return Set_Mal
procedure Normalize(Set_1, V_1, V_2)
    avg_2 ← 0
    if Σ_{d∈Set_1} V_good[d] > 0 then avg_2 ← Σ_{d∈Set_1} V_2[d] / |{d∈Set_1 : V_2[d]>0}|
    for x ∈ Domains do
        if V_1[x] > 1 then V_1[x] ← 1
        if x ∈ Set_1 then V_2[x] ← 0
        if V_1[x] > 1 − avg_2 then V_1[x] ← 1
```

a vertex $v \in BC$, connected only to vertices in BC

$$((M^T) * V)[v] = const_{BC} \cdot \sum_{v \neq i \in BC} V[i] \tag{1}$$

Due to space limitation, the proof of this theorem is omitted.

Using the property of a balanced clique we devised the following algorithm for computing the scores. The reputation of every clique vertex is the sum of reputation scores of all other vertices of the clique, multiplied by the constant edge weight of the clique. This is shown in algorithm 4.

Algorithm 4 AssignBCScores

```
procedure AssignBCScores(BC, const_BC, V)
    Sum ← Σ_{i∈BC} V[i]
    for i ∈ BC do
        V_result[i] = (Sum − V[i]) · const_BC
    return V_result[i]
```

The complexity of this algorithm is $O(|BC|)$ which is a significant improvement to $O(|BC|^2)$, the complexity of the matrix multiplication approach. In our

graph, all edges between the same type of vertices (IPs or domains) belong to balanced cliques and therefore this optimization plays a major factor.

4 Experiment results

The evaluation of the algorithm uses real data collected from several sources. To understand the experiments and the results, we first describe the data obtained for constructing the graph, and the criteria used for evaluating the results.

4.1 Data sources

We used five sources of data to construct the graph.

- A-records: a database of successful mappings between IPs and domains, collected by Cyren [7] from a large ISP over several months. This data consists of over one milion domains and IPs which are used to construct the nodes of the graph.
- Feed-framework: a list of malicious domains collected and analyzed by Cyren over the same period of time as the collected A-records. This list is intersected with the domains that appeared in the A-records and serves as the initial known "bad" domains vector.
- Whois [15]: a query and response protocol that is widely used for querying databases that store the registered users or assigners of an Internet resource, such as a domain name, an IP address block, or an autonomous system. We use WHOIS to get the IP data, which consists of the five characteristics of IP (ASN, BGP prefix, registrar, country, registration date).
- VirusTotal [14] - a website that provides scanning of domains for viruses and other malware. It uses information from 52 different antivirus products and provides the time a malware domain was detected by one of them.
- Alexa: Alexa database ranks websites based on a combined measure of page views and distinct site users. Alexa lists the "top websites" based on this data averaged over a three-months period. We use the set of top domains as our initial benign domains, intersecting it with the domains in the A-records. This set is filtered to remove domains which appeared as malicious in VirusTotal.

We conducted two sets of experiments *Tuning-test* and *Time-test* . In the *Tuning-test* experiment, the DNS graph is built from the entire set of A-records, but the domains obtained by the Feed-Framework are divided into two subsets: an *Initial set* and a *Test set*. The Initial set is used as the initial vector of "bad" domains for the flow algorithm. The test set is left out to be identified as bad by the algorithm. Obviously, not all bad domains of the test set can be identified, mainly because the division of domains was done randomly and some of the domains in the test set are not connected to the initial set. Yet, this experiment was very useful to determine the best parameters of the algorithm which were used later in the Time-Test experiment. The *time-test* experiment, is carried out

in two steps corresponding to two consecutive time periods. In the first step, we construct the graph from the data sources described above. We use the feed-framework data to set the initial vector of bad domains in the first time period and the Alexa data to set the initial vector of good domains. We execute the flow algorithm (combined or extended) and assign the final score for each node of the graph.

To validate the results of the Time-test, we check the domains detected as malicious by the algorithm against data from VirusTotal for the period following the time period used in the first step. We sort the domains by descending bad reputation score and use the score of the domain on the k-th position as the threshold. As the evaluation criteria for this test, we define a *Prediction Factor (PF)* which evaluate the ability of our algorithm to detect bad domains identified later by VirusTotal. We compute this factor as the ratio of domains labeled as bad that were tagged by VirusTotal and a random set of domains of the same size, that were tagged by VirusTotal.

A domain tested with VirusTotal, is considered as tagged only if it was tagged by at least one anti-virus program. A PF_i factor considers a domain as tagged by VirusTotal if at least i anti viruses tagged the domain.

To compute the prediction factor we extract the set of domains with the k-highest bad reputation scores $HSet_k$ found by the algorithm and select randomly a set of k domains $RSet_k$:

$$PF_i = \frac{|HSet_k \cap Set^i_{tagged}|}{|RSet_k \cap Set^i_{tagged}|} \tag{2}$$

where set^i_{tagged} is the set of all domains tagged by at least i anti viruses.

A value of PF_i indicates the extent to which our algorithm identifies correctly bad domains, compared to the performance of randomly selected domains. A similar approach was used by Cohen et al. [5] that compared the precision of a list of suspicious accounts returned by a spam detector against a randomly generated list. Next we describe the results of the two experiments.

4.2 The Tuning test

There are two main objectives to the Tuning test. The first is to verify that domains with higher bad reputation values are more likely to be detected as bad domains. The second objective is to understand the impact of the different parameters on the final results and configure the evaluation experiment accordingly. We tested the different weight functions discussed in section 3.1. The following combination was found best and was selected to compute the weights on the graph edges:

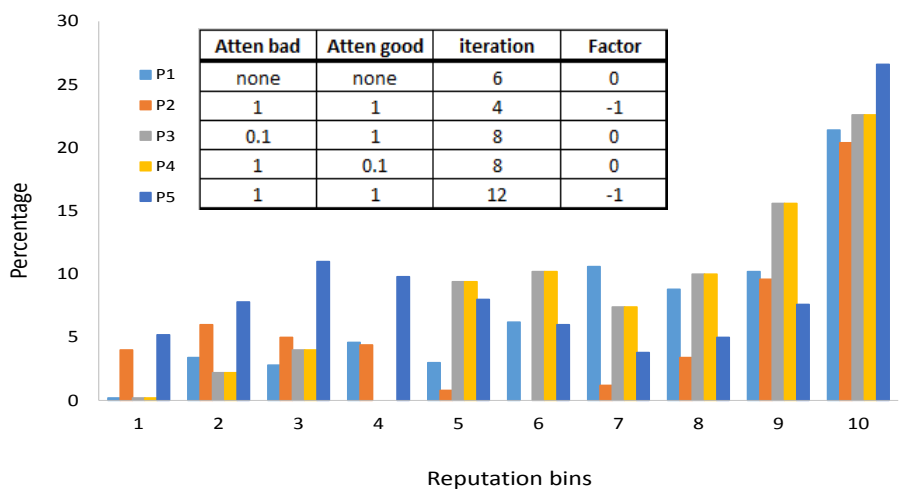
- IP to domain - for $ip \in Set_{IP}$ and $d \in D_{ip}$, $w(ip, d) = \frac{1}{|D_{ip}|}$.
- Domain to IP - for $d \in Set_{domain}$ and $ip \in I_d$, $w(d, ip) = \frac{1}{|I_d|}$.
- IP to IP - for $ip_1, ip_2 \in Set_{commonAtt}$, s.t. $ip_2 \neq ip_1$, $w(ip_1, ip_2) = \frac{1}{|Set_{commonAtt}|}$.

– Domain to Domain - for $d_1, d_2 \in Set_{domain}$, parents $P_d \subset Set_{parent}$, and $d_1, d2 \in P_d$, s.t. $d_1 \neq d_2$, $w(d_1, d_2) = \frac{1}{|P_d|}$.

The most effective combination of features for IP data turned to be (none, ASN, BGP-Prefix, Registrar, none), which derives cliques of IPs with the same ASN, BGP-Prefix and Registrar.

The tuning test also examines the attenuation of the flow, the number of iterations, and the relative importance of the good domains used to calculate the final score. The tuning test repeats the following steps:

1. Construct the graph from the entire A-records database.
2. Divide the set of 2580 tagged malicious domains from the feed-framework into two sets: an initial set of 2080 domains and a test set consisting of 500 domains.
3. Apply the flow algorithm (combined or extended) using the Initial set to initiate the vector of bad domains and the data from Alexa to initiate the vector of good (as described above).



Atten bad	Atten good	iteration	Factor
none	none	6	0
1	1	4	-1
0.1	1	8	0
1	0.1	8	0
1	1	12	-1

Fig. 2: The percentage of malicious domains by reputation score

We repeat these steps with different combinations of parameters (see Figure 2). We then compute the reputation scores of all domains, sort the domains by these scores and divide them into 10 bins of equal size. For each bin S_i, $i = 1..10$ we measure the presence of domains that belong to the test set S_{test} (known malicious domains), as $\frac{|S_i \cap S_{test}|}{|S_{test}|}$.

In Figure 2 we can see that the 10% of domains with the highest 'bad' reputation score contain the highest amount of malicious domains. Moreover, in

all five type of experiments, the bin of domains with the highest scores consists of 20-25 percents of all known malicious domains of S_{test} .

The tests we conducted demonstrate the following three claims:

1. Domains with high 'bad' scores have a higher probability to be malicious.
2. The combinations of the "good" and "bad" reputation scores improve the results.
3. After a relatively small number of iterations the results converge.

4.3 Time-test

The time test experiment uses the best parameters as determined by the Tuning test. The first step of the experiment uses data collected during three months: September-November 2014 while the second step uses data collected during the following two months (i.e., December 2014- January 2015). After the first step in which we apply the flow algorithm (either the combined or the extended), and assign bad reputation scores to domains, we validate the domains with the highest 'bad' reputation scores that did not appear in the initial set of malicious domains against the information provided by VirusTotal.

The data we used for the time test consists of $2,385,432$ Domains and $933,001$ IPs constructing the nodes of the graph, from which 956 are tagged as maliciouse and $1,322$ are tagged as benign (Alexa). The resulting edges are $2,814,963$ Ip to Domain edges, $13,052,917,356$ Domain to Domain edges and $289,097,046$ IP to IP edges. The very large numbers of edges between IP to IP and Domain to Domain, emphasize the importance of the optimized algorithm 4 in Section 3.3. For the validation test we selected the 1000-highest bad reputation domains and calculated the prediction factor PF_i for $i \in \{1, 2\}$ (see equation 2).

Out of 1000 random domains checked against VirusTotal, only 2.775% were tagged by at least one anti virus product and 0.875% of the random domains were tagged by at least two anti virus product.

Table 1: Results for time test experiment

	Atten bad	Atten good	Algorithm	Tagged by one	Tagged by two
1	0.8	1	combined	334	212
2	1	1	combined	323	203
3	none	none	extended	247	136

Table 1 presents the results of the experiment using the Combined and Extended algorithms. We used only 5 iterations since the Tuning test results indicates that this number is sufficient. For the combined algorithm the weight of good reputation score was set to $w = -1$ and the bad attenuation was set to 1 and 0.8 in tests 1 and 2 respectively.

We can see from the table that with the parameters of P2, that from the 1000 domains with the highest score, 323 were tagged by at least one of the anti viruses in VirusTotal (tagged by one), and 203 were tagged by at least two anti viruses (tagged by two) which derive the prediction factor of $PF_1 = 11.61$ and $PF_2 = 23.2$, respectively.

Due to space limitations we do not present all the results, however the best results were achieved in the second test. For example, out of 200 domains with the highest scores in test 2, 107 domains were tagged which reflect 53.5% of all known malicious domains.

Fig. 3: Prediction factor with respect to the number of highest bad domains

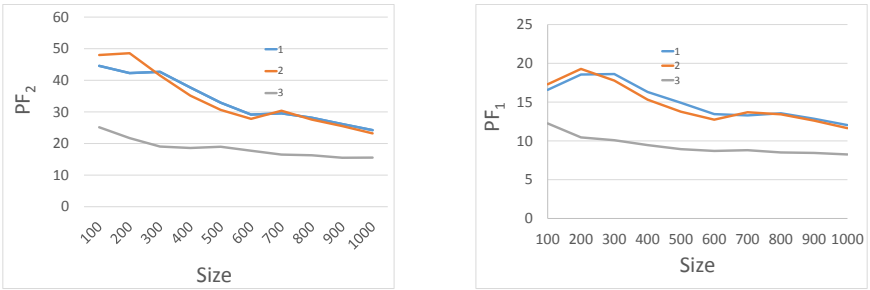


Figure 3 demonstrates the ability of our algorithms to predict malicious domains, using the prediction factor (eq. 2). The figure shows that if we consider smaller size of domains with the highest score the prediction rate is better but the overall predicted malicious domains are less. If we consider a larger size we end up with a larger number of domains that we suspect as malicious by mistake(i.e., false positives).

5 Conclusions

This paper discusses the problem of detecting unknown malicious domains by estimating their reputation score. It applies a classical flow algorithms for propagating trust, on a DNS topology graph database, for computing reputation of domains and thus discovering new suspicious domains. Our work faces two major challenges: the first is building a DNS graph to represent connections between IPs and domains based on the network topology only and not on the dynamic behavior. The second challenge was the design and implementation of a flow algorithm similar to those used for computing trust and distrust between people. We presented an algorithm that can be deployed easily by an ISP using non private DNS records. The algorithm was evaluated to demonstrate its effectiveness in real-life situations. The results presented in this paper are quite

encouraging. In future work we intend to further improve the results by taking more data features into consideration and by conducting further tests with more parameters.

Acknowledgement This research was supported by a grant from the Israeli Ministry of Economics, The Israeli Ministry of Science , Technology and Space and the Computer Science Frankel center.

References

1. Alexa: Websites Ranking . https://www.alexa.com/ (2014)
2. Antonakakis, M., Perdisc, R., Dagon, D., Lee, W., Feamster, N.: Building a dynamic reputation model for dns. In: USENIX Security Symposium. pp. 273–290 (2010)
3. Antonakakis, M., Perdisci, R., Le, W.: Detecting malware domains at the upper dns hierarchy. In: USENIX Security Symposium (2011)
4. Choi, H., Lee, H.: Identifying botnets by capturing group activities in dns traffic. In: Computer Networks 56. pp. 20–33 (2012)
5. Cohen, Y., Gordon, D., Hendler, D.: Early detection of outgoing spammers in large-scale service provider networks. In: DIMVA. pp. 83–101 (2013)
6. Coskun, B., Dietrich, S., Memon, N.D.: Friends of an enemy: identifying local members of peer-to-peer botnets using mutual contacts. In: ACSAC. pp. 131–140 (2010)
7. Cyren: A provider of cloud-based security solutions . https://www.cyren.com/ (2014)
8. D.Kamvar, S., Schlosser, M.T., GarciaMolina, H.: The eigentrust algorithm for reputation management in p2p networks. In: WWW. pp. 640–651 (2003)
9. Guha, R., Kumar, R., Raghavan, P., Tomkins, A.: Propagation of trust and distrus. In: WWW. pp. 403–412 (2004)
10. Leyla, B., Engin, K., Christopher, K., Marco, B.: Exposure finding malicious domains using passive dns analysis. In: NDSS (2011)
11. Mui, L., Mohtashemi, M., Halberstadt, A.: A computational model of trust and reputation for e-businesses. In: Proceedings of the 35th Annual Hawaii International Conference on System Sciences $HICSS02$. vol. 7, p. 188. Washington, DC, USA (2002)
12. Page, L., S. Brin, R.M., Winograd, T.: Pagerank citation ranking: Bringing order to the web. In: Technical report, Stanford Digital Library Technologies Projext. (1998)
13. Villamarin-Salomon, R., Brustolon, J.C.: Bayesian bot detection based on dns traffic similarity. In: SAC. pp. 2035–2041 (2009)
14. VirusTotal: A free virus, malware and URL online scanning service. https://www.virustotal.com/ (2014)
15. Whois: IP data. who.is (2014)
16. Witten, I., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniquel. Morgan Kaufmann (2005)
17. Wu, B., Goel, V., Davison, B.D.: Propagating trust and distrust to demote web spam. In: WWW Workshop on Models of Trust for the Web and MTW (2006)
18. Yadav, S., Reddy, A.K.K., Reddy, A.L.N., Ranjan, S.: Detecting algorithmically generated malicious domain names. In: Internet Measurement Conference. pp. 48–61 (2010)