Advanced Flow Models for Computing the Reputation of Internet Domains

Hussien Othman[1], Ehud Gudes[1], and Nurit Gal-Oz[2](✉)

[1] Ben-Gurion University, Beer-Sheva 84105, Israel
{hussien,ehud}@cs.bgu.ac.il
[2] Sapir Academic College, D.N Hof Ashkelon, 79165 Ashkelon, 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. We believe that a domain that is related to malicious domains is more likely to be malicious as well and therefore detecting malicious domains using the DNS network topology is a key challenge.

In this work we improve the flow model presented by Mishsky et al. [12] for computing the reputation of domains. This flow model is applied on a graph of domains and IPs and propagates their reputation scores through the edges that connect them to express the impact of malicious domains on related domains. We propose the use of clustering to guide the flow of reputation in the graph and examine two different clustering methods to identify groups of domains and IPs that are strongly related. The flow algorithms use these groups to emphasize the influence of nodes within the same cluster on each other. We evaluate the algorithms using a large database received from a commercial company. The experimental evaluation of our work have shown the expected improvement over previous work [12] in detecting malicious domains.

1 Introduction

Domain reputation has become an essential tool in fighting advanced malware and Advanced Persistent Threats (APT). Since detecting sophisticated malware in real-time is difficult, the use of domain reputation is a key objective for companies like Dambella [8] or Cyren [7] which provide services for identifying threats imposed by malicious domains and IPs.

The common approach to assess domains is to compute features from DNS records and queries responses, and use these features to train a classifier that labels domains as malicious or benign. This approach is effective as long as the attackers do not manipulate these features. However, DNS features are not robust [16], since the attackers can change the features of malicious domains to evade detection. For example, they can change Time To Live (TTL) for DNS queries, patterns in domain names, etc.

J.-P. Steghöfer and B. Esfandiari (Eds.): IFIPTM 2017, IFIP AICT 505, pp. 119–134, 2017.
DOI: 10.1007/978-3-319-59171-1_10

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. FluxBuster [14], is a passive DNS traffic analysis tool for detecting malicious flux networks. Using DNS mapping data, Perdisci et al. [14] apply an hierarchical clustering on the domains where the distance between two domains is calculated according to the number of mutual resolved IPs, so that each of these clusters represents a candidate flux network. This work motivated our clustering based approach.

In this paper we extend the flow model presented by Mishsky et al. [12] for calculating the reputation scores of domains. Flow models are used in trust based reputation systems such as EigenTrust [11] and Pagerank ([13]). The assumption underlying the work of Mishsky et al. [12] is that IPs and domains which are neighbors of malware-generating IPs and domains, are more likely to become malware-generating as well. The outline of their approach is as follows. Domains and IPs are represented as nodes in a directed weighted graph. The nodes in the graph are initially assigned a reputation value based on two lists of labeled domains: a list of 'bad' domains which is compiled from various internet databases (e.g., VirusTotal [19]) and a list of 'good' domains based on the Alexa database [1]. A major challenge in building the graph is the assignment of weights to edges to reflect the strength of relation between the nodes they connect. Their approach uses the statistical features which are associated with mappings of domains and IPs. Starting with this graph, a flow algorithm is applied to propagate the initial reputation scores from each node to its neighbors in an iterative manner so that the impact of malicious nodes on their neighbors is presented. The evaluation of this model [12] demonstrates its ability to identify malicious domains and the experimental results on real-life data are quite impressive.

The main contribution of the current work is the improvement of previous work [12] in two steps. We first extend the graph by using new attributes which are related to the registration information of domains and name servers hosting them. Next we propose the clustering approach to strengthen weights on edges between domains and IPs that seem to be highly correlated. We examine two clustering methods, Categorical and Communities. Categorical clustering groups domains based on their mutual attributes, while Communities clustering groups domains and IPs based on their mutual relations.

The rest of this paper is organized as follows: in Sect. 2 we present the related work and discuss briefly the work of [12] which is our starting point. In Sects. 3 and 4 we present our contribution to improve this work, starting with the new attributes we propose; we explain our clustering methods and the respective flow algorithms. We demonstrate the results of our experimental evaluation in Sect. 5, and in Sect. 6 we conclude and discuss future work.

2 Related Work

There are quite a few papers which use DNS data logs to detect Botnets and malicious domains.

Most of the papers use the DNS traffic behavior, rather than mapping information. Some recent papers such as Notos [2] and FluxBuster [14] use the mapping information in order to compute a reputation score for malicious domains. However, both require a huge amount of mappings between IPs and Domains in order to succeed. For example, in [14] false positives and true negatives in the experiments are explained by the fact that some domains are not mapped to enough IPs. Villamarin-Salomon et al. [18] provide C&C (Command and Control) detection technique motivated by the fact that bots typically initiate contact with C&C servers to poll for instructions. For each domain, they aggregate the number of non-existent domains (NXDOMAIN) responses per hour (denoted NX-during-hour-i) and also compute the query rate of each second level domain (SLD) per hour. They use the average and standard variation of these rates as features to detect abnormally high or temporally concentrated rates. Based on these features, a vector of a certain domain is classified as anomalous using the Mahalanobis Distance metric [9]. The suspects produced by this system were also independently reported as suspicious by other detectors. The Exposure system [3] collects data from 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 [22] in order to determine whether a domain name is malicious or not.

Choi et al. [5] use passively monitored DNS traffic to detect botnets, which form a group activity in similar DNS queries simultaneously. They assume that infected hosts (bots) perform DNS queries at the following occasions: successful host infection, malicious attack DDoS attack, Spam mailing, C&C server link failures (repetitive attempts), server migration (to communicate new site location) and IP address changes. Using this data they construct a feature vector and perform clustering to identify the malicious domains. Their proposed technique can detect botnets irrespective of their communication protocols and C&C server migration however it faces the problem of obtaining DNS traffic data (compared to the use of DNS topological data which is much easier to obtain). Another recent paper by Pedrici et al. [15] uses traffic data and behavior based traffic from which a traffic graph is built and analyzed.

2.1 Starting Point: A Topology Based Flow Model

Mishsky et al. [12] address the problem of detecting unknown malicious domains by estimating their reputation score. A classical flow algorithm for propagating trust is applied on a DNS topology graph database, for computing reputation of domains and thus discovering new suspicious domains. This model uses DNS IP-Domain mappings and statistical information but does not use DNS traffic data as done by others (e.g., [2]). 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. In [12] a graph is constructed to reflect the topology of domains and IPs and their relationships and a flow model is applied to propagate the knowledge received in the form of black-list, to label domains in the graph as suspected domains. Domains and IPs are represented as nodes in a directed weighted graph while the edges of the graph describe their network connections. The two types of vertices in the topology graph, domains and IPs, derive four types of edges between them: Domain-Domain, IP-IP, Domain-IP and IP-Domain. Two domains are connected in the graph if they have in common a mutual parent. Two IPs are connected if they have the same ASN, BGP, registrar, and date attributes. A Domain is connected to an IP (or IP to Domain), if the IP was resolved for that domain according to A-Records (a database of successful mappings between IPs and domains). To construct real-life graphs data collected from an ISP and from *WHOIS* database [21] is used. The graph nodes are initially labeled based on a list of malicious domains which is compiled from various Internet databases (e.g., VirusTotal [19]). Weights are assigned to the graph edges based on statistical features which are associated with the domains and IP mappings. The flow algorithm applied on this graph, propagates reputation from one node to another while discounting the strength of the connection between them as expressed by the weight on the edge connecting them. The results of the flow algorithm, which is similar to Eigentrust [11] are used to indicate the extent to which nodes are suspected to be malicious.

3 DNS Topology Graph Extension

To construct the DNS topology graph we use additional attributes to those used by Mishsky et al. [12], that better determine the significance of the connections between the nodes. The new features are based on the assumption that malicious domains reuse valuable resources. Most popular attacks depend on the availability of many resources which are often purchased [20]. For example: domain names are registered or transferred for a price, large numbers of infected hosts are available for rent, bullet-proof servers are available for rent [20], etc. Moreover, many types of resources are made to be reusable so that they can be resold multiple times to maximize financial gain. The reuse of resources across different attacks may reveal connections between malicious domains.
In this work we investigate the use of three types of resource attributes:

Domain's parent: A $k - Top$ level domain ($kTLD$) is the k suffix of the domain name. Following [2] for each domain we define k parents as $iTLD$ for $i = 1..k$. 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*. This attribute was used by [12]. In our experiments we restrict the parents to $2TLD$.

Registration: Registrant name, Address, Email, Organization and Registrar. These attributes are important since the WHOIS information [21] about a malicious domain sometimes includes certain pseudo-identity details such as the same/similar fake registrant name, the same registrant email, same registrant address, etc [20].

NameServer: The motivation for using name server relation is that one name server can provide the DNS records for a large number of malicious domains.

Similar to [12], we define a weight function that assigns a weight to each edge in the graph. There are two types of vertices in the topology graph: domains and IPs, deriving four types of edges between them. Let Set_{IP} denotes the set of all the IPs and let Set_{domain} denotes the set of all the domains.

Let $Attributes_{domains}$ be the set of attributes we use in the graph to define the relation between domains. $Attributes_{domains} =$ (registrant name, registrant city, registrant country, registrant email provider, registrant organization, registrar, name server, parent).

Let Set_{α} be the set of all domains which have in common attribute α where $\alpha \in Attributes_{domains}$.

Let w be a weight function $w : (v, u) \rightarrow [0, 1]$ used to assign weight to the edge (u, v) where $u \in Set_{IP}$ or $u \in Set_{domain}$ and $v \in Set_{IP}$ or $v \in Set_{domain}$.

For edges of type *domain-ip, ip-ip, ip-domain* we use the same formula as in [12].

- Weights on *domain-ip edges:* 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: $w(d, ip) = \frac{1}{|I_d|}$.
- Weights on *ip-domain edges:* 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: $w(ip, d) = \frac{1}{|D_{ip}|}$.
- Weights on *ip-ip edges:* Let $commonAtt$ be a combination of ASN, BGP, registrar, date attributes. Let $Set_{commonAtt}$ be the set of all IPs with the attribute combination $commonAtt$. For each $ip_1, ip_2 \in Set_{commonAtt}$ s.t. $ip_1 \neq ip_2$ we define: $w(ip_1, ip_2) = \frac{1}{|Set_{commonAtt}|}$.
- Weights on *Domain-Domain edges*: here we use the new attributes defined above. Let Set_{α} be the set of all domains with the same attribute α as domain d. For each $d1, d2 \in Set_{\alpha}$ s.t. $d1 \neq d2$ we define the following weight metric: $w_{\alpha}(d_1, d_2) = \frac{1}{|Set_{\alpha}|}$. We define $w(d_1, d_2)$ as average of all the w_{α} for $\alpha \in Attributes_{domains}$.

4 Clustering Based Flow Model

The new attributes we add to distill the significance of the connections between the nodes, improve the method of [12], but not to the extent expected (see Sect. 5.3). A possible reason for this is the relatively low flow received by nodes of low centrality according to [12]. To overcome this problem we use clustering to classify domains. By clustering we identify groups of domains that are related to each other and direct the flow accordingly.

We use two types of clustering, *Categorical Clustering* which involves domain nodes only, and *Communities Clustering* which involves both domains and IPs. We refer to the latter as communities. For each type of clustering we have a Flow model that uses only relevant edges.

4.1 Categorical Clustering

Categorical clustering identifies groups of strongly connected domains where the strength of connection is determined by the number of attributes they share with each other. We define v_d, the vector of attributes for domain d as follows:

$v_d =$ (Registrar, Registrant name, Registrant email provider, Registrant city, Registrant country, Registrant organization, Name Server, Parent).

In the categorical algorithm, we use K-Modes [10] that extends the k-means paradigm to cluster categorical data. Following the K-Modes algorithm we estimate the strength of connection between any two domains, by calculating the dissimilarity distance between their attribute vectors.

The **clustering based flow Algorithm**, described in Algorithm 1, is different from the flow algorithm presented in Mishsky et al. [12]. In Algorithm 1 the edges are restricted to two types: IP to IP edges and Domain to IP or IP to Domain edges. The edges between domains are removed, and the propagation of flow from domain to another domain is done by dividing the average score in the cluster to all the domains it includes. In each iteration, we propagate the flow between IPs and Domains. Only at the end of the iteration we propagate flow within each cluster of domains.

Algorithm 1 starts with the initialization phase in lines 23–30 in which the score of each domain is set to the average score of all the domains in its cluster. These scores are passed as parameter to the algorithm as $V_{initial}$. In each iteration of the flow (lines 6–19), we propagate scores from domain to IP and IP to domain (lines 6–12) and from IP to IP (lines 13–19).

At the end of each iteration (line 18), we compute the new score of each cluster and propagate the increment gained in the iteration to all the domains in the cluster. The flow algorithm is executed separately by Algorithm 2 to propagate benign reputation and malicious reputation. It calls Algorithm 1 with an initial set of domains that are labeled either malicious or benign.

Applying the algorithm separately to propagate malicious and benign reputation may result in a node that is labeled both benign and malicious. The final labeling of such node depends on the relative importance given to each label as done in Algorithm 2. The parameters of this algorithm are two vectors of domains' reputation scores, one with their reputation as benign and the other with their reputation as malicious. The initial score of a domain as malicious is set in the malicious vector to 1 if it appears in the black lists (obtained by various Internet databases e.g., VirusTotal [19]) and 0 otherwise. The reputation score of domains in the benign vector is set the same way using the benign domains as obtained from Alexa [1]. In line 6 the scores are combined, and using a threshold of θ we label the domains as malicious.

4.2 Communities of IPs and Domains

In contrast to clustering of domains only, the community based clustering uses the entire generated graph including IPs, Domains and the weighted edges. Our goal is to examine whether malicious domains and IPs are grouped together in

Algorithm 1. Flow with clustering

```
1: procedure ClusterBasedFlow($Vector\ V_{initial}$, $Iterations\ n$)
2:     for $C \in Clusters$ do
3:         $InitializeScores(C, V_{initial})$
4:     end for
5:     for 1 to $n$ do
6:         for $e = (d, ip) \in Set_{IP-Domain-Edges} \vee e = (ip, d) \in Set_{Domain-IP-Edges}$ do
7:             $prevScore_d = score_d$
8:             $score_d += weight_e * score_{ip}$
9:             $score_{ip} += weight_e * prevScore_d$
10:            $C = GetCluster(d)$
11:            $C.increment += score_d - prevScore_d$
12:        end for
13:        for $e = (ip_1, ip_2) \in Set_{IP-IP-Edges}$ do
14:            $prevScore_{ip_1} = score_{ip_1}$
15:            $score_{ip_1} += weight_e * score_{ip_2}$
16:            $score_{ip_2} += weight_e * prevScore_{ip_1}$
17:        end for
18:        for $C \in Clusters$ do
19:            $propagateScore(C)$
20:        end for
21:    end for
22: end procedure
23: procedure InitializeScores($Cluster\ C$, $Vector\ V_{initial}$)
24:    $totalScore = 0$
25:    for $d \in C$ do
26:        $totalScore = totalScore + V_{initial}(d)$
27:    end for
28:    for $d \in C$ do
29:        $score_d = V_{initial}(d) + \frac{totalScore}{C.size}$
30:    end for
31: end procedure
32: procedure propagateScore(Cluster C)
33:    for $d \in C$ do
34:        $score_d = score_d + \frac{C.increment}{C.size}$
35:    end for
36: end procedure
37: procedure GetCluster(Domain $d$)
38:    for $C \in Clusters$ do
39:        if $d \in C$ then
40:            return $C$
41:        end if
42:    end for
43: end procedure
```

clusters we call communities. The community detection algorithm we use follows the Louvain Method for community detection [4]. In our graph, domains and IPs are connected to each other via multiple relation measures. Domains are connected to each other according to their parent domain, registrant information and name server serving them. IPs are connected by ASN, registrar,

Algorithm 2. Combine benign and malicious flow (n, V_{benign}, V_{mal})
Input
n: number of iterations, V_{benign}: a vector of benign domains, V_{mal}: a vector of malicious domains

```
1: $V_{benign} \leftarrow ClusterBasedFlow(V_{benign}, n)$
2: $V_{mal} \leftarrow ClusterBasedFlow(V_{mal}, n)$
3: $Set_{Mal} \leftarrow \emptyset$
4: for $d \in Set_{Domain}$ do
5:     if $V_{mal}[d] + W_{benign} \cdot V_{benign}[d] > \theta$ then $Set_{Mal} \leftarrow Set_{Mal} \cup \{d\}$
6:     end if
7: end for
8: return $Set_{Mal}$
```

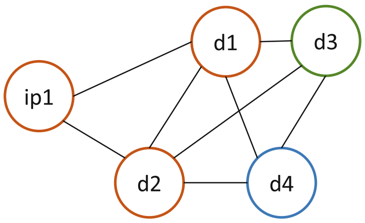

Fig. 1. Community graph example: the score propagated from d_1 to d_3 and d_4 is weaker than the score propagated from d_1 to d_2 which share the same IP $ip1$.

BGP prefix, date. IPs and Domains are connected by A-records mapping. By clustering we divide our graph to communities where each community contains IPs and domains that have the strongest connections to each other. To avoid false positives, we build a new weighting function for the flow algorithm based on communities. For example in Fig. 1 all the domains have in common a mutual parent, registrant information and name server, hence we add edges between each pair. However, d_1 and d_2 have in common a mutual IP, ip_1. Assuming d_1 is malicious, in the previous method, the domains d_2, d_3, d_4 will get equally bad score from d_1 that will be propagated to their neighbors. Using communities we may have d_1, d_2 in the same cluster due to their connection to ip_1 while d_3, d_4 remain out of the cluster, hence they will receive a smaller bad score from d_1. This way we refine the propagation and make a more careful distinction.

The community clustering begins with a graph of IPs and Domains as nodes and the weighted edges connecting them. We use the weights defined in Sect. 3 with one exception: for IP-Domain and Domain-IP edges, we multiply the weight by a factor of α, $\alpha > 1$. The aim of this factor is to strengthen the connection between IP and Domain so it will not be discarded by the Louvain algorithm. Since in our graph there are significantly greater number of edges between domains than edges between IP and Domain, domains are more likely to be added to a community that contains Domains with same mutual attributes and discard the IP connection. We prevent this by adding the α factor to these edges. Once we have clustered the graph to communities we use the community-based flow algorithm to compute the reputation of each domain.

The **Communities-based flow algorithm**, uses two different flow models, within a community and between different communities. These models differ in the weighting scheme they use:

- *Flow inside a community:* The weight of each edge (u, v) where $u \in C_i$ and $v \in C_i$ is calculated using the same weight function as defined in Sect. 3 considering only the induced subgraph constructed by all the nodes of C_i.
- *Flow across communities:* we allow flow between communities. Hence, we could not treat each community as a separated graph. Rather, we treat it all as one big graph and apply a new weighting scheme. The new weighting scheme applies to any type of edge (IP-IP, IP-Domian, Domian-IP, Domain-Domain). Let v denote a vertex in the graph. An *inside edge* (u, v) where $u \in C_i$ and $v \in C_i$ is an edge connecting two vertices inside the community. An *outside edge* (u, v) where $u \in C_i$ and $v \notin C_i$ is an edge connecting a vertex inside the community with a vertex outside the community. Let n_1 denote the number of inside edges connecting v. Let n_2 denote the number of outside edges connecting v. For each inside edge we assign the weight: $\frac{1}{n_1 x}$.

  For each outside edge we assign the weight: $\frac{1}{(n_1+n_2)y}$.

  For each vertex v, the sum of all the weights on the edges connected to it is 1, therefore: $\frac{n_1}{n_1 x} + \frac{n_2}{(n_1+n_2)y} = 1$.

  Let $f = \frac{y}{x}$, the weight of each inside edge:

$$\frac{(n_1+n_2)f}{(f+1)n_2 n_1 + f{n_1}^2} \tag{1}$$

  The weight of each outside edge:

$$\frac{1}{(f+1)n_2 + f n_1} \tag{2}$$

  The ratio between the weights is

$$\frac{inside}{outside} = \frac{f(n_1+n_2)}{n_1} \tag{3}$$

  If n_2 is much greater than n_1 (most of the edges are outside edges) the weight inside is much greater than outside. If n_1 is much greater than n_2 (most of the edges are inside edges), the weights are almost equal (where f is close to 1). We therefore use f to tune the ratio between weights inside and outside a community, and examine this parameter in our experiments.

5 Evaluation

The evaluation of our methods uses real data collected from several sources. We first describe our data and the criteria obtained for evaluating the results.

5.1 Data Sources

We use the following 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 million domains and IPs which are used to construct the graph nodes.
– 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 malicious domains vector.
– WHOIS [21]: 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) and to get the registrant information on the domains.
– VirustTotal [19]: a website that provides scanning of domains for viruses and other malware. It uses information from 52 different anti-virus products and provides the time that a malware domain was detected by one of them.
– Alexa [1]: 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.

Table 1 presents the size and the characteristics of the data we use to construct our graph.

After constructing the graph each experiment is conducted twice, once with initial malicious domains and once with initial benign domains. The score of each domain is computed (those who did not receive any flow remain with score zero). The scores are sorted and compared to Virus Total scans.

Table 1. Data characteristics

Vertices	Number of domains	1007833
	Number of IPs	345451
	Number of Malicious domains	462
Edges	Number of edges from IP To Domain	1116823
	Number of edges from Domain To Domain	119055774
	Number of edges from IP To IP	29260535

5.2 Evaluation Criteria

In the real world one would prefer to minimize the risk and get a list of suspicious domains even if only a small part of them is malicious. In many cases the cost of checking suspicious benign domains is worth the risk of getting attacked by a real malware. Our aim is to identify suspicious domains to reduce the amount of unnecessary checks.

We run Algorithm 1 with the lists of malicious domains and with the list of benign domains separately and obtain two vectors: $Vector_{mal}$ - representing the malicious reputation of domains and $Vector_{benign}$ - representing the benign reputation of domains. The flow algorithm assigns reputation scores to all domains. Using $W_{benign} \in [-1, 0]$, to discount the benign measure, the score of a domain d is computed according to Algorithm 2 for combining benign and malicious flow:

$$score[d] = Vector_{mal}[d] + W_{benign} \cdot Vector_{benign}[d] \tag{4}$$

We sort the domains by descending reputation score and use the score of the domain on the $k-th$ position as the threshold, denoted θ. The group of domains with score higher than the threshold are selected, denoted $HSet_k$: $HSet_k = \{d \in Domains | score[d] \geq \theta\}$.

We compare this set to the ground truth which is available from VirusTotal [19] to obtain the domains in $HSet_k$ that are correctly tagged as malicious denoted $GTSet_k$. Let $MalSet$ denote the set of malicious domains according to VirusTotal [19], then $GTSet_k = \{d \in HSet_k | d \in MalSet\}$. The evaluation criteria is the ratio of domains that were correctly identified as malicious out of the set $HSet_k$. The *TRatio* criteria representing the precision of positive prediction, is calculated as follows:

$$TRatio = \frac{|GTSet_k|}{|k|} \tag{5}$$

A domain tested with VirusTotal [19], is considered as tagged only if it was tagged by at least two anti-virus programs. A similar approach was used by Cohen et al. [6] to compare the precision of a list of suspicious accounts returned by a Spam detector against a randomly generated list.

5.3 Experiment Results

In this section we analyze the results of the proposed clustering methods and compare them to the results of the original method [12] we attempt to improve. For categorical clustering we use the k-modes clustering algorithm to construct clusters of domains which have common attributes. We evaluate the results with respect to the number of clusters generated by the k-modes algorithm as shown in Fig. 2.

When the number of clusters (k) is between 500 and 1000 we identify the largest number of malicious domains. As the number of clusters increases and the average cluster size decreases accordingly, less malicious domains are identified since less flow enters the cluster from outside. On the other hand, smaller values

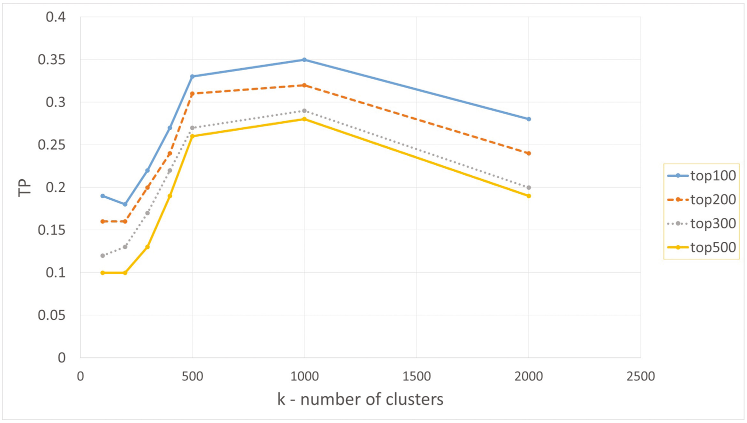

Fig. 2. Categorical Clustering: precision with respect to the number of clusters

of k which result in larger clusters introduce too much noise and too many false positives. We have verified against VirusTotal [19] about 35% of the top 100 scores and 28% of the top 500 scores.

We compare the results of the categorical clustering to the original methods of [12] by testing all algorithms with the data described in Sect. 5.1. Table 2 presents the improvement of the methods proposed in this paper in two steps. First by introducing the new attributes, and second by using both attributes and clustering. However it is important to note that our dataset is smaller and less populated than the one used in [12].

Table 2. Comparison of the results with the original method [12]

Method	Top100	Top200	Top300	Top500
[12]	10%	10%	9.6%	9.2%
[12] + New attributes	20%	17.5%	16.6%	16.2%
[12] + New attributes + Categorical clustering ($k = 1000$)	35%	32%	29%	28%

In the second test, we divide our graph into communities using the Louvain algorithm as described in Sect. 4, where each community expresses better correlation between its domains and IPs. To prevent the Louvain algorithm from discarding the connections between Domains and IPs, we use the α factor as described in Sect. 4. The best results for our graph were obtained with $\alpha = 100$. Figure 4 shows the impact of each common attribute on the community clustering. The figure shows that domains with common attributes do not necessarily belong to the same community. For example, on the average only 0.18 of pairs of domains that have the same Registrar attribute belong to the same community.

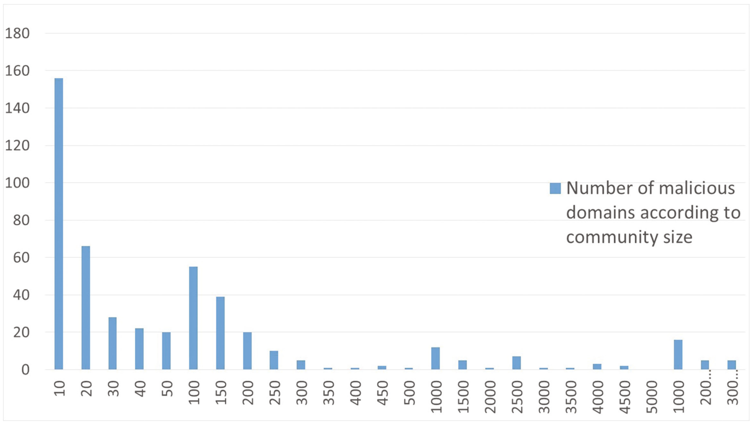

Fig. 3. Distribution of malicious domains in communities with respect to community size

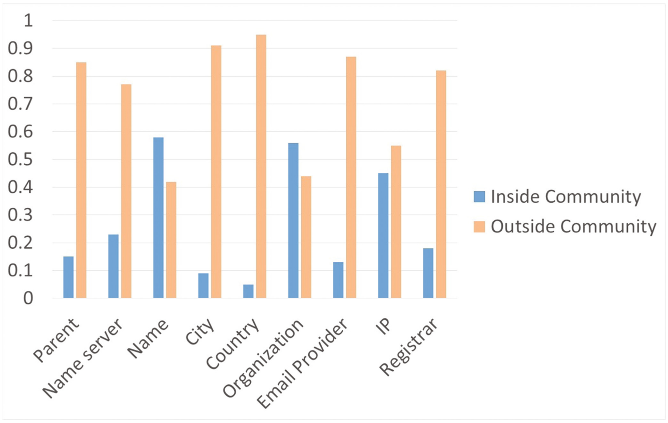

Fig. 4. Average distribution of mutual attribute inside and outside the community.

Only 0.45 of the IPs belong to the community of the domains to which they are mapped.

Figure 3 depicts the size of communities in which malicious domains reside. For example, about 160 malicious domains reside in communities of size smaller than 10.

Flow inside communities: We set the weights on the edges as described in Sect. 4 and run the flow algorithm. Table 3 depicts the percentage of malicious domains detected out of the top rated. The results present a significant improvement over previous experiments. We have identified 68 malicious domains among the top 100 scores, and 101 malicious domains among the top 200 scores. Furthermore, 51 out of 65 top rated malicious domains where identified which represent

Table 3. Flow inside communities results

Top100	Top200	Top300	Top500
68%	50.5%	33.6%	20.2%

Table 4. Flow across communities results

Top100	Top200	Top300	Top500
42%	39%	39%	38%

about 78% detection. Table 3 also shows that when considering more than the 200 top rated malicious domains, the precision decreases and less domains are detected. The reason is the restriction of flow inside the communities only. There is no propagation of malicious score from malicious domains to communities which don't contain malicious domains initially. Therefore even if malicious domains reside in these communities, they are not detected by the flow algorithm. Roughly only about 5% of the graph could be reached via flow inside communities which contain malicious domains.

Flow across communities: In this experiment we allow flow between communities so that a domain's score is affected by both, domains inside its community and related domains that reside outside its community. The best results are shown in Table 4. The results of the flow inside communities are better for the top 100 and 200 scores but for top 300 scores and further the flow across communities performs much better. This demonstrates the limitation of the flow inside communities, which is unable to find the malicious domains identified by the second method.

Figure 5 depicts the results obtained with respect to the f parameter defined in Sect. 4 for tuning the ratio between weights of inside-edges and outside-edges. We show the results of the experiments where f varies between 0.1 and 5. The best results are obtained for higher values of f in this range, where the weights inside communities are increased and outside communities are decreased.

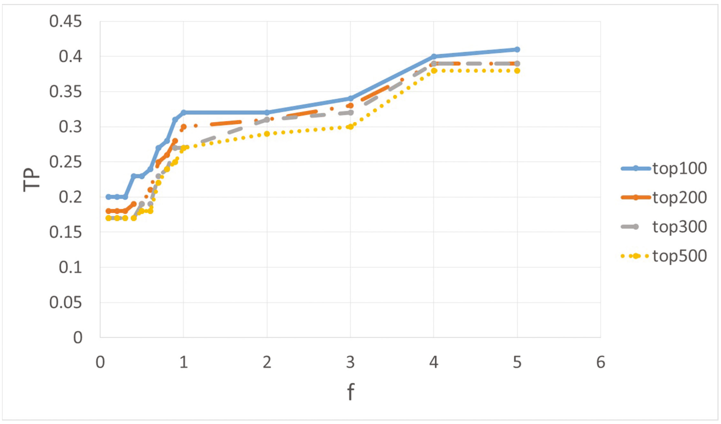

Fig. 5. Flow across communities: precision with respect to f parameter

However, in experiments with larger values of f where the relative importance of outside edges becomes very small, the behavior is similar to the behavior of flow inside communities only.

From a security point of view, the flow across community can be used as a means to identify more malicious domains at the price of reduced precision.

6 Conclusions and Future Work

Computing accurately domain reputation is an important factor in preventing the spread of malware by malicious domains. The use of Trust flow for computing domain reputation was first reported in [12]. In this paper we improve the results of [12] using new attributes and a clustering based flow model. We have presented two types of clustering. The first type, categorical clustering, involves clustering of domains only and is based on multiple attributes. The improvement demonstrated by this algorithm was minor. The second type, communities clustering, involves clustering of both domains and IPs by using the concept of Communities. This clustering further improves the results. The best results in the top 100 scores were achieved from flow inside communities only (close to 80%), in which we restricted the flow to domains and IPs residing in the same community. However, in that model we didn't get enough "bad" scores to evaluate the top 500 scores. The other communities based model yield quite good results with respect to top 100 and top 500 scores, while identifying almost 40% malicious domains in the top 500 scores. This is an improvement over [12] which found only 30% malicious domains.

In future work we intend to extend our work in two directions. The first is to combine our flow model with classification models (e.g., [2]) to overcome situations where statistical features are not good enough to distinct between malicious and benign domains (for example, domains that are resolved to small set of IP addresses). The other direction is to integrate the results of profiling methods used for anomaly detection [17], with the flow model. Based on behavioral attributes an anomaly score can be used to identify malicious domains and the relation between reputation and anomaly score can be examined.

Acknowledgement. This research was supported in part by the Lynn and William Frankel Center for Computer Sciences at Ben-Gurion University, Israel, and we like to thank them for their support. We also thank the reviewers for very useful comments.

References

1. Alexa. The web information company (2014). https://www.alexa.com/
2. 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)
3. Bilge, L., Kirda, E., Kruegel, C., Balduzzi, M.: Exposure: finding malicious domains using passive DNS analysis. In: NDSS (2011)

4. Blondel, V.D., Guillaume, J.-L., Lambiotte, R., Lefebvre, E.: Fast unfolding of community hierarchies in large networks. CoRR, abs/0803.0476 (2008)
5. Choi, H., Lee, H.: Identifying botnets by capturing group activities in dns traffic. Comput. Netw. **56**(1), 20–33 (2012)
6. Cohen, Y., Gordon, D., Hendler, D.: Early detection of outgoing spammers in large-scale service provider networks. In: Rieck, K., Stewin, P., Seifert, J.-P. (eds.) DIMVA 2013. LNCS, vol. 7967, pp. 83–101. Springer, Heidelberg (2013). doi:10.1007/978-3-642-39235-1_5
7. Cyren. The web information company (2016). http://www.cyren.com/
8. Dambella. The web information company (2016). https://www.damballa.com
9. De Maesschalck, R., Jouan-Rimbaud, D., Massart, D.L.: The mahalanobis distance. Chemometr. Intell. Lab. Syst. **50**(1), 1–18 (2000)
10. Huang, Z.: Extensions to the k-means algorithm for clustering large data sets with categorical values. Data Min. Knowl. Discov. **2**(3), 283–304 (1998)
11. Kamvar, S.D., Schlosser, M.T., Garcia-Molina, H.: The Eigentrust algorithm for reputation management in p2p networks. In: Proceedings of the 12th International Conference on World Wide Web, pp. 640–651. ACM (2003)
12. Mishsky, I., Gal-Oz, N., Gudes, E.: A topology based flow model for computing domain reputation. In: Samarati, P. (ed.) DBSec 2015. LNCS, vol. 9149, pp. 277–292. Springer, Cham (2015). doi:10.1007/978-3-319-20810-7_20
13. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web (1999)
14. Perdisci, R., Corona, I., Giacinto, G.: Early detection of malicious flux networks via large-scale passive DNS traffic analysis. IEEE Trans. Dependable Sec. Comput. **9**(5), 714–726 (2012)
15. Rahbarinia, B., Perdisci, R., Antonakakis, M.: Segugio: efficient behavior-based tracking of malware-control domains in large ISP networks. In: 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2015, Rio de Janeiro, Brazil, 22–25 June 2015, pp. 403–414 (2015)
16. Stinson, E., Mitchell, J.C.: Towards systematic evaluation of the evadability of bot/botnet detection methods. In: Proceedings of the 2nd Conference on USENIX Workshop on Offensive Technologies, WOOT 2008, Berkeley, CA, USA, pp. 5:1–5:9. USENIX Association (2008)
17. Villamarín-Salomón, R., Brustoloni, J.C.: Identifying botnets using anomaly detection techniques applied to DNS traffic. In: 2008 5th IEEE Consumer Communications and Networking Conference, CCNC 2008, pp. 476–481. IEEE (2008)
18. Villamarín-Salomón, R., Brustoloni, J.C.: Bayesian bot detection based on DNS traffic similarity. In: Proceedings of the 2009 ACM Symposium on Applied Computing, pp. 2035–2041. ACM (2009)
19. VirusTotal. A free virus, malware and URL online scanning service (2014). https://www.virustotal.com/
20. Xu, W., Sanders, K., Zhang, Y.: We know it before you do: predicting malicious domains. In: Virus Bulletin Conference (2014)
21. Whois. IP data (2014). https://who.is/
22. Witten, I.H., Frank, E.: Data Mining: Practical Machine Learning Tools and Techniques. Morgan Kaufmann, San Francisco (2005)