Systematic Mining of Associated Server Herds for Malware Campaign Discovery

Jialong Zhang
Texas A&M University
jialong@cse.tamu.edu

Sabyasachi Saha*
Symantec
saby_saha@symantec.com

Guofei Gu
Texas A&M University
guofei@cse.tamu.edu

Sung-Ju Lee*
KAIST
sjlee@cs.kaist.ac.kr

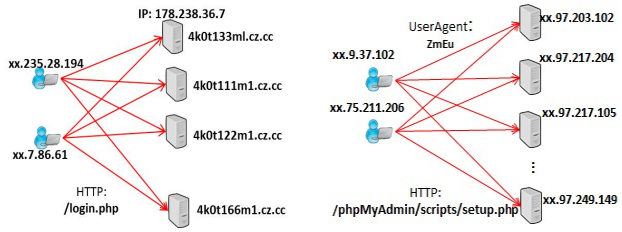
Marco Mellia
Politecnico de Torino
mellia@tlc.polito.it

***Abstract*—HTTP is a popular channel for malware to communicate with malicious servers (e.g., Command & Control, drive-by download, drop-zone), as well as to attack benign servers. By utilizing HTTP requests, malware easily disguises itself under a large amount of benign HTTP traffic. Thus, identifying malicious HTTP activities is challenging. We leverage an insight that cyber criminals are increasingly using dynamic malicious infrastructures with *multiple* servers to be efficient and anonymous in (i) malware distribution (using redirectors and exploit servers), (ii) control (using C&C servers) and (iii) monetization (using payment servers), and (iv) being robust against server takedowns (using multiple backups for each type of servers). Instead of focusing on detecting *individual* malicious domains, we propose a complementary approach to identify a *group* of closely related servers that are potentially involved in the same malware campaign, which we term as *Associated Server Herd (ASH)*. Our solution, SMASH (Systematic Mining of Associated Server Herds), utilizes an unsupervised framework to infer malware ASHs by systematically mining the relations among all servers from multiple dimensions. We build a prototype system of SMASH and evaluate it with traces from a large ISP. The result shows that SMASH successfully infers a large number of previously undetected malicious servers and possible zero-day attacks, with low false positives. We believe the inferred ASHs provide a better global view of the attack campaign that may not be easily captured by detecting only individual servers.**

I. Introduction

Malware is a critical security threat to the Internet. Malware is increasingly using HTTP as its communication and attacking channel due to the following reasons: (i) HTTP is allowed in most networks, and thus malware has a good chance to infect victims and communicate with attackers. (ii) Since HTTP now is the majority of network traffic, malware can easily disguise its activities under huge benign HTTP traffic, making it difficult to be detected. (iii) Most HTTP requests use domain names to find servers, thus malware can easily evade IP blocking or hide their servers by using Fast-Flux [25]. As a result, 75 % of malware is observed to generate HTTP traffic [28], and the number of Web-based attacks increased by almost three-fold since 2012 [9].

Given the severity and popularity of HTTP based malware, we focus on such threats. In particular, we concentrate on detecting malicious HTTP activities from the *server side communication* perspective.[1] We define two types of malicious activities: (i) communication activity and (ii) attacking activity. Communication activity involves malware's communication with *malicious servers*, while attacking activity involves malware's attack on *benign servers*. We define a malicious campaign as a set of servers that are involved in these malicious activities. We observe that malicious campaigns in both types of malicious activities typically share very similar server-side properties. Figure 1 shows real examples. In the communication activity (Figure 1(a)), there are two clients sending HTTP requests to multiple C&C domains. Malware often uses such domain fluxing to evade detection, leading to sharing the same IP address. In addition, since these C&C servers use the same communication protocol, they utilize the same script "login.php" to handle the requests from their bots. These multiples C&C domains form a malicious communication campaign. In the attacking activity (Figure 1(b)), which is a ZmEu scanning campaign, there are two clients/bots that kept scanning seven benign servers targeting on "setup.php" script, which has a known code injection vulnerability. In this case, those two clients/bots scan the default path of phpMyAdmin for the exploitation leading to sharing the same file "setup.php". Those seven targeted servers form a malicious attacking campaign.



(a) Communication activity (b) Attacking activity

Figure 1. Malicious campaigns examples.

Many approaches have been proposed to detect malware activities. Different from existing work that relies on signatures [27], client side behavior patterns [21], or supervised learning of individual server reputation [17], we propose an *unsupervised* approach that focuses on *server side* communication patterns and does *not* rely on signatures. We leverage an insight that cyber criminals are increasingly using a dynamic malicious infrastructure with *multiple* servers to be

*Work done while working at Narus Inc.

[1]By servers, we mean both IP addresses and domain names.

efficient and anonymous in (i) malware distribution (using redirectors and exploit servers), (ii) control (using C&C servers), (iii) monetization (using payment servers), and (iv) being robust against server takedown (using multiple backups for each type of servers). As a result, in each malware campaign, there are multiple malware servers used as well as common benign servers they attacked. As illustrated in Figure 1, these servers are correlated, e.g., they share similar client sets. This is typically not true for benign servers because different (independent) servers usually have different sets of clients.[2]

This insight comes from an inquisition that benign servers usually serve different benign users whose behaviors might be diverse while malicious servers are set up for certain malicious clients. Not only connected by a similar set of clients, but also if these servers are the same type (e.g., both are exploit or C&C servers), they are likely to receive requests targeting the same/similar URI files (e.g., vulnerable files or exploit scripts) from malware clients. For benign servers, each server usually has lots of scripts/pages and different users likely visit different pages for different purposes. On the other hand, as malicious servers are set up for certain purpose (e.g., C&C, malware downloading), it only uses certain scripts/pages to handle all their bots' requests. In addition, we observe many other correlation among malware servers in the same campaign. For example, they may have the same IP address although with many different domain names (i.e., domain-fluxing, as shown in Figure 1(a)), or their domains are registered by the same organization at a similar time.

Based on the above insights, instead of focusing on detecting *individual* malicious domains, we propose a complementary approach to identify a *group* of closely related servers that are involved in the same malware campaign, which we term as *Associated Server Herd (ASH)*. Our scheme, SMASH (Systematic Mining of Associated Server Herds), is designed to be deployed at enterprise or ISP networks to automatically detect malicious servers that communicate with their bot/malware armies. It uses an unsupervised community detection technique to characterize the relationship among the servers from multiple dimensions, *e.g.* if they are contacted by common clients, if the same or similar files are accessed/downloaded from them, or if they have the same Whois information, etc. Our data mining based approach exposes that often servers involved in an attack retain some similarity at multiple dimensions and we can detect such groups by combining them. Therefore, SMASH is not a real-time detection system, however, it can be run everyday to detect daily malicious activities in a large ISP/Enterprise networks or be run on a large network traffic trace to dig out previously unknown malicious activities.

[2]Even in the case of load balancing or Content Distribution Networks where multiple benign servers are used, these servers are likely to serve different set of clients (e.g., based on their locations).

SMASH detects malicious campaigns by correlating ASHs generated from multiple dimensions. Although each dimension itself might not be sufficient to distinguish malicious severs from benign servers, the combination of these dimensions can generate ASHs involved in malicious campaigns. The suspicious score of correlated ASHs is based on different combinations. The more close relationship an ASH has, the higher the probability the servers in it are involved in malicious activities.

Our main contributions are summarized as follows:

- We propose SMASH, a system that detects a variety of attacking campaigns and malicious communication campaigns using an unsupervised data mining approach. Since our approach is unsupervised, it can detect zero-day malware campaigns.
- We propose a two-step method to identify groups of servers involved in a malware campaign. In the first step, we generate multiple ASHs using graph based clustering on individual dimensions. In the second step, we detect malicious ASHs by correlating them. Rather than detecting a single server in isolation, SMASH infers ASHs by looking at the global, holistic network view. Moreover, by correlating in multiple dimensions, SMASH is robust to manipulation and evasion from attackers.
- We evaluate SMASH with 9 days of large ISP data and present the details of the malicious campaigns it discovered. SMASH detected servers involved in both attacking and communication campaigns with the highest false positive rate of only 0.064%. SMASH found a total of 236 confirmed malicious campaigns with more than 10,000 servers involved in malicious activities. SMASH discovered nearly 7× the number of servers detected by a commercial IDS and blacklists. SMASH also revealed 600 benign servers suffering from web injection campaigns while IDS detected only four. Other examples of inferred real world campaigns include Bagle botnet, Conficker botnet, Zeus botnet, Sality botnet, TDSS botnet, etc.

II. Related Work

Detecting malicious domains has been widely studied from different angles. Many schemes detect malicious domains from the DNS point of view. In [15, 17], the authors used different features (e.g., number of distinct TLDs, number of distinct malware samples that contacted the domain, changes in the number of requests to a domain) to evaluate the reputation of each single domain *in isolation*. However, such methods can not detect servers involved in attacking activities, and need malicious domain seeds to train their system. In [16], Antonkakis et. al. focused only on malicious domains generated by DGA malware, and such method can not be applied to general malicious domains. Kopis [15] can be used to detect general malicious domains. However, it needs to monitor DNS traffic at the upper DNS hierarchy, which dramatically limits its application.

Another line of research detects malicious domains by extracting signatures from malware traffic and applying generated signatures to live network to detect malware traffic [27, 28]. Perdisci et al. [28] proposed a system that clusters malware samples requesting similar URLs and generates structure signatures from them. The generated signatures can be used to detect infected hosts on live networks. SMASH targets both malicious IP addresses and domains that are involved in attacking and communication activities. In addition, our system is an completely unsupervised system that does not need malicious traffic seeds to train features or build templates/signatures.

Gao et al. [20] studied the temporal relationship among servers to infer malicious domains that always appear with the seed malicious domains. Li et al. [26] studied the topology among malicious servers and inferred other malicious servers from a small set of malicious severs seeds. However, the limitation of these propagation based detection systems is that their effectiveness depends on the malicious seeds, thus can not be used to detect servers involved in new, unknown activities that malicious seeds can not cover.

Community/clustering based techniques have also been widely researched in both spam and malware detection. Zhang et al. [31] proposed to detect comment spam by exploiting the relationship among benign servers that are targeted by comment spammers. Recently, Invernizzi et al. [24] proposed a system to detect malware distribution networks. It explored four techniques to group candidate connections and built a neighbor graph to further filter false positives. Stringhini et al. [29] detected malicious web pages by focusing on HTTP redirections. Their approach first grouped URLs based on the different combination of TLD, Domain, Pages, IP and Parameters, and further classified malicious groups with 28 features. All above systems either target on specific attack channels (e.g., Forum spam, HTTP redirections and malware distribution network) or require a large and diverse user base [29], which limits their practicality. In contrast, SMASH targets on more generic malicious servers and with fewer prerequisites.

Gu et al. [21, 22] proposed anomaly-based botnet detection systems that look for similar network behaviors across client hosts. A set of bots that share similar anomaly patterns are detected as botnets. Yen et al. [30] detected malware by aggregating traffic that share the same external destinations or similar payload, and involve internal hosts with similar OS platforms. The intuition behind these work is that hosts infected with the same bot malware usually have common C&C communication patterns. Therefore, they infer the infected clients by analyzing the relationship among clients. Different from these work, SMASH focuses on malicious servers; we study the relations among servers because server-side infrastructure is more robust and stable. While malware can easily randomize client-side traffic patterns (e.g., injecting random content in their packets, sending requests to random benign websites), they inevitably need to contact their malicious servers to fulfill their desired functions. In addition, client based approaches usually require multiple infections of clients in a network. We believe SMASH is an excellent complementary system to client-side based detection systems.

III. System Design

The primary goal of SMASH is to detect suspicious correlated servers that are involved in malicious activities by passively looking at the network-wide HTTP communications. Such malicious activities include launching HTTP attacks on benign severs and communicating with malicious servers through the HTTP channel. Instead of detecting each server in isolation, we study the *different relationship* among all the servers involved in similar activities. Those servers involved in the same malicious activity are inferred as a malicious campaign by SMASH.

Figure 2 depicts the architecture of SMASH. The system takes HTTP network traffic as input, and has five components: traffic preprocessing, ASH mining, ASH correlation, pruning, and malicious campaign inference.

A. Preprocessing

The goal of preprocessing is to reduce the traffic that need to be processed by SMASH. We explore two steps to reduce the number of input servers to SMASH. First, we assume that domains with the same second-level domain belong to the same organization.[3] For example, a.xyx.com.cn and b.xyz.com.cn both belong to xyz.com.cn, thus there is no need to differentiate them. Some CDN/Cloud servers will be also aggregated as one server in this case. For example, all the Facebook CDN servers will be aggregated as "fbcdn.net". Amazon cloud servers will be aggregated as "amazonaws.com". The aggregation of all domains based on their second-level domains leads to 60% reduction of all servers.

We further remove most benign servers based on their popularity.[4] To measure the "popularity" of servers, we utilize the concept of inverse document frequency (IDF), which is a measure of whether the term is common across all documents in information retrieval. In our case, we try to remove common servers across all the clients' requests. We define the popularity of a server as the number of clients that communicated with the server. The more clients the server is connected to, the more popular the server is.

We select an IDF threshold of 200 based on the popularity distribution of IDS confirmed malicious servers, which filters

[3]There are some exceptions such as cloud servers, dynamic DNS. We will discuss them in Section VI.

[4]We acknowledge that we may miss some compromised popular domains. However, we argue that this represents a necessary tradeoff between performance and accuracy. In reality most popular servers have resources and incentives to secure their websites and thus have a lower possibility to be compromised than less popular ones.

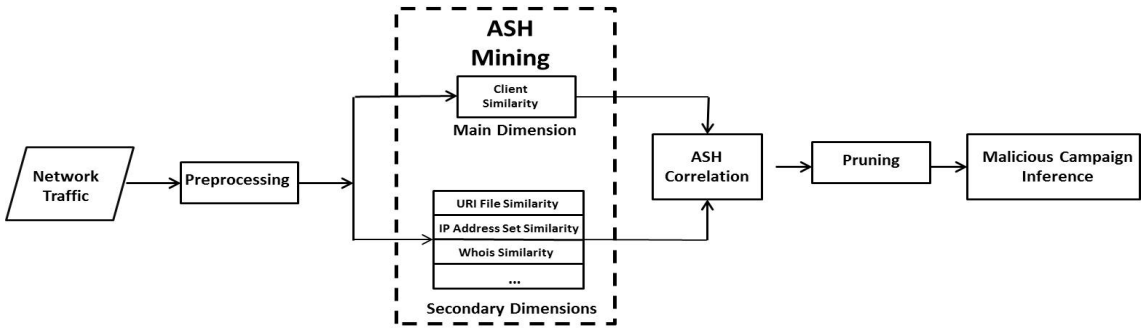


Figure 2. System overview.

very popular servers but still keeps 99% of the servers. After the preprocessing process, we reduced 58.6% of traffic in our dataset.

B. Associated Server Herd Mining

The goal of ASH mining is to find closely related servers that are involved in the same malware campaign. We define one main dimension and three secondary dimensions to characterize the relationship among the servers, and systematically mine ASHs. ASH generated from each dimension itself might not be sufficient to distinguish malicious group of servers from benign servers, but ASHs associated with the combination of these dimensions are more likely to generate server groups involved in malicious campaigns.

To find the correlated servers, a simple way is to assign each server with a feature vector and perform clustering on this multi-dimension feature vector. However, the dimensions are different in nature and it is inefficient to combine them to evaluate similarity. Also, as we show in Section V-C, it is hard to assign a unique weight for each dimension because different malicious campaigns rely on different combinations of those dimensions. We observe that malicious servers in the same malicious campaign usually share a very similar (if not the same) set of malware clients while those malicious servers are usually not connected by benign clients. Thus, we use client similarity as the main dimension, which is much more robust against manipulation from attackers than other dimensions, and can reliably group the servers.

We see that although client similarity alone may not directly distinguish malicious servers from benign servers, it separates benign server groups from malicious server groups. Thus, the main dimension must be satisfied for all campaigns.

Each secondary dimension characterizes the relationship among different servers from a certain perspective. We evaluate how their combinations can be used to infer malicious campaigns in Section V-C. Note that we envision SMASH, as an extensible system, can easily incorporate new dimensions. For example, to keep our system lightweight, we have not included all payload downloaded from each host. However, this can be an interesting dimension to consider and can be easily added as another dimension.

1) Main Dimension: We use *client similarity* as the main dimension. Client similarity between two servers depends on the common set of clients contacting them. We define the client similarity between servers S_i and S_j as:

$$Client(S_i, S_j) = \frac{|C_{s_i} \bigcap C_{s_j}|}{|C_{s_i}|} * \frac{|C_{s_j} \bigcap C_{s_i}|}{|C_{s_j}|} \qquad (1)$$

where C_{S_i} denotes the set of clients contacting server S_i. The ratio $\frac{|C_{S_i} \bigcap C_{S_j}|}{|C_{S_i}|}$ represents the importance of the common clients for server S_i. The intuition here is that if two servers with many clients are similar, there will be large overlap between their clients. Thus, two servers are similar when their common clients are important to both servers.

Since malicious servers are usually not connected by benign clients while infected clients are usually connected to a same set of suspicious servers, two servers sharing similar sets of client connections should belong to the same ASH. Specifically, we build a communication graph $G = (V, E)$, where V denotes all the servers and each edge $(i, j) \in E$ denotes that servers i and j share a set of clients. The weight of the edges reflects the strength of similarity between the two servers in terms of client similarity.

To extract ASH from G, we adopt a graph based clustering algorithm [18] that is designed to efficiently uncover communities in large networks. It uses modularity to measure the quality of extracted community, which is a scalar value between -1 and 1, and represents the density of the links inside the community as compared with the links between communities. It automatically finds high modularity partitions of large networks in short time. The nodes that are still connected to each other after this process form ASHs of the main dimension.

2) Secondary Dimensions: We present our current secondary dimensions.

URI File Similarity: We study the relationship among servers based on URI files as servers in the same malicious activities might share similar/same URI files. For example,

jikdooyt0.com | /waX3dj1X5L3juWu3dmVyPTQuMCZiaWQ9YjZjYWVhNjE0NjhhMmQ4ZTc0OGQ3ZTEzMTIyMDZiMDQ4NWY2MjJhYSZhaWQ9NDAxOTcmc2lkPTAmcmQ9MCZlbmc9d3d3Lmdvb2dsZS5pdCZxPXF1aWZ1bWV0dGk=06x

skolewcho.com | /cVu4PVle8W4M1mc3dmVyPTQuMCZiaWQ9YjZjYWVhNjE0NjhhMmQ4ZTc0OGQ3ZTEzMTIyMDZiMDQ4NWY2MjJhYSZhaWQ9NDAxOTcmc2lkPTAmcmQ9MCZlbmc9d3d3Lmdvb2dsZS5pdCZxPWNhc3RyYXppb24=37k

Figure 3. Obfuscated filenames.

web attacks target certain vulnerable files, and thus different targeted servers share the same destination files. Different C&C servers in the same campaign may use the same scripts to handle the requests from the infected clients, and hence they might also share the same files. We extract all the URI files of the servers by checking the HTTP requests. Here we focus on URI files rather than the whole URI path because in attacking activities, some benign servers share the same vulnerable file but have different paths due to the different configurations on each web server.

We define a URI file as the substring of a URI starting from the last '/' until the end before the question mark, which usually is the file or script used for handling clients' requests. As shown in Figure 3, sometimes attackers use obfuscated filenames for different malicious servers that are involved in the same malicious campaign. We define URI file similarity between the two files as follows. If the length of the filename is shorter than or equal to len, we define the similarity function of files f_i and f_j as:

$$sim(f_i, f_j) = \begin{cases} 1 & if f_i = f_j, \quad (2) \\ 0 & otherwise. \quad (3) \end{cases}$$

Thus, two files are similar if they are exactly the same, since short filenames are usually not obfuscated. However, if the length of a filename is longer than len[5], we define the similarity function as:

$$sim(f_i, f_j) = \begin{cases} 1 & if \cos(\theta) > 0.8 \quad (4) \\ 0 & otherwise \quad (5) \end{cases}$$

where

$$\cos(\theta) = \frac{CharSet_{f_i} \cdot Charset_{f_j}}{\|CharSet_{f_i}\| \cdot \|CharSet_{f_j}\|}. \quad (6)$$

where $CharSet_{f_i}$ is the character distribution vector of file name f_i. Thus, for long filenames, we check the characters frequency distribution ($CharSet$ in Eq. (6)) of the filenames. Two filenames are similar as long as their names have similar character distributions. For the exact same filenames, the similarity score is 1. While our similarity function works well in our evaluation, it can be replaced by any similarity functions such as Levenshtein distance and Hamming Distance.

[5] len is empirically set to 25 in this paper, which is based on the length distribution of the filenames whose servers have been labeled by IDS.

jikdooyt0.com | Moniker Privacy Services | 1800 SW 1st Avenue | +1.5032070147 | NS1.ABOVE.COM | NS2.ABOVE.COM

skolewcho.com | BARONOFDOMAINS | 1800 SW 1st Avenue | +1.5032070147 | NS1.ABOVE.COM | NS2.ABOVE.COM

Figure 4. Whois similarity.

We now define the file similarity between the two servers S_i and S_j as

$$File(S_i, S_j) = \frac{\sum_m max_n(sim(f_{S_{im}}, f_{S_{jn}}))}{\sum_m 1} * \frac{\sum_n max_m(sim(f_{S_{jn}}, f_{S_{im}}))}{\sum_n 1} \quad (7)$$

where $f_{S_{im}}$ is the m-th file from server S_i. Similar to client similarity, the first term of the right hand side of Eq. (7) reflects the importance of similar files to server S_i, and the second term of the right hand side of the equation reflects the importance of similar files to server S_j. Thus, if two servers share enough similar files, they might be involved in the same activities, and should be in the same ASH.

IP Address Set Similarity: We investigate the relationship among the servers based on their IP addresses because malicious domains may share a similar set of IP addresses. For example, malicious servers may use fast flux to evade domain based detection, and thus multiple domains may share the same IP address. In our dataset, skolewcho.com, switcho81.com, jikdooty0.com and swltch081.com all used the same IP address. Similar to client similarity, we define IP address set similarity as:

$$IP(S_i, S_j) = \frac{|I_{S_i} \bigcap I_{S_j}|}{|I_{S_i}|} * \frac{|I_{S_j} \bigcap I_{S_i}|}{|I_{S_j}|} \quad (8)$$

where I_{S_i} is the set of IPs that server S_i is associated with. Thus, if two servers share similar IP addresses, they might be involved in the same activities and should be in the same ASH.

Whois Similarity: We study the relationship among servers based on their whois information as malicious servers may be registered with similar information, such as register name, home address, email address, phone number and name servers. Figure 4 shows the whois information of two example malicious servers. Although they have different registrants, they share the same home address, phone number and name servers. We use the whois similarity to measure the relationship among servers, which is defined as the number of shared fields between the two servers over the union of fields. We require that the two servers share at least two above mentioned fields to be considered as associated servers to avoid the case that two servers only share the domain name registration proxy.

3) ASH Generation: After studying the similarity among servers from different multiple dimensions, we build similarity graphs for different dimensions, and use the same graph based community detection algorithm mentioned in Section III-B1 to generate ASHs. The nodes connected to

each other after this process form ASHs for each dimension.

C. Associated Server Herd Correlation

Once we obtain the ASHs from different dimensions, we perform ASH correlation. The goal of multi-dimension correlation is to distinguish malicious ASHs from benign ASHs. To achieve this, we consider ASHs in different dimensions and extract their common associated servers to form new ASHs. Ideally, the more dimensions an ASH belongs to, the more likely it is involved in malicious activities. The intersection of ASHs between the main dimension and secondary dimensions forms the new suspicious ASHs.

For example, $(ASH_j^d \bigcap ASH_i^m)$ forms a new ASH combining ASH_i^m from the main dimension m and ASH_j^d from a secondary dimension d. We compute the suspicious score for each server in the new ASH as follows:

$$S(S_i) = \sum_{d \in Sec\_Dimensions} w_d(C_{S_i}^d) w_m(C_{S_i}^m) \Phi(|C_{S_i}^d \bigcap C_{S_i}^m|) \tag{9}$$

where $\Phi(x) = \frac{1}{2}(1 + erf(\frac{x-\mu}{\gamma}))$, $erf(\cdot)$ is the "S" shaped Gaussian error function, and μ and γ are user specified parameters.[6] $C_{S_i}^m$ is an ASH from dimension m that includes server S_i and $w_d(C_{S_i}^d)$ represents the ASH density. Density is measured as the number of edges $|e|$ in one group over the number of edges in the fully connected graph with $|v|$ vertices in that group $(2 * |e|/(|v| * (|v| - 1))$. The intuition here is that the more dense a group is, the more likely it belongs to a malicious group, as benign servers are less likely to be well connected. When we obtain the suspicious score for each server in the newly formed ASH, the servers whose scores are below the threshold $thresh$ are removed. In addition, the groups with only one server left are also removed because that server can not be associated with others. We discuss the selection of $thresh$ value in Section V.

In Eq. (9), $\Phi(ASH_i^m \bigcap ASH_j^d)$ measures the suspiciousness of the newly formed ASH, created with the servers common in two ASHs formed based on dimensions m and d. It is based on the size of the ASH and promotes the ASHs with a large number of servers. A smaller value of $|ASH_i^m \bigcap ASH_j^d|$ means that there are only few servers in the ASH, and we have less confidence in its maliciousness. Hence we need to cross check with more dimensions to make a decision.

The "S" shaped $\Phi()$ normalizes $\Phi(ASH_i^m \bigcap ASH_j^d)$ into a value between 0 and 1. After the normalization, a group with less than four servers receives a low score, and need to be cross checked with more dimensions to accumulate higher suspicious scores. For each dimension, the highest score is 1. In this case, the correlation score reflects how the ASHs are formed. Suspiciousness score $S(.)$ of each server then accumulates scores from all ASHs it belongs to. The higher the score, the more suspicious the server is. If a server has suspiciousness score below $thresh$, it gets removed from all the ASHs. After the removal, the ASHs (created by combining multiple dimensions) are left with only servers with high scores. For example, a score higher than 1.0 means that the server is inferred through one main dimension and at least two secondary dimensions. We then call the ASHs with high scoring servers as suspicious.

[6]We empirically set $\mu = 4$ and $\gamma = 5.5$. Choice of μ promotes the clusters with size larger than 4. γ determines the desired steepness of the curve.

D. Pruning

After the correlation, we define and prune two types of noisy ASHs: (i) Redirection Group and (ii) Referrer Group. For the redirection group, some servers are associated with each other because they belong to the same redirection chain. Hence they share exactly the same sets of clients, IP addresses, and sometimes URI files. For the referrer group, some servers are associated with each other because they are referred by the same landing server (e.g, landing websites are embedded with other websites).

To remove these noisy servers without eliminating malicious servers, we use their landing servers to replace all the servers in the same redirection chain instead of simply dropping those groups, if all the servers in the chain share same IP addresses, URI files or Whois information. Similar to the redirection group, we also use landing servers to replace all the referred servers. The intuition here is that for the redirection and the referrer groups, if a client visits the landing server, it *automatically* visits other servers in the redirection chain or the embedded servers. We therefore use only the landing server to represent those servers.

We collect the redirection chains by sending a HTTP request to each server in the ASHs, and obtain referrer information by extracting the HTTP "referrer" field from the input network traffic. After the pruning process, if there still exist more than one server in the ASH, we keep that group as a candidate malicious ASH.

E. Malicious Campaign Inference

ASH correlation process typically captures specific malicious activities, but not the whole malicious activities. For example, bots first download encrypted files from some servers and then connect to other C&C servers. In this case, ASH correlation process might separate these two processes into two different herds, making it difficult to analyze the file downloading activities. Towards this end, we apply a refinement step in which we rebuild the original attack campaign based on the client similarity. Two malicious ASHs are merged when their servers are in the same herd for the main dimension, i.e., they share a very similar set of clients. The intuition is that the main dimension captures the group connection behaviors of malicious activities, and the infected clients that connect to different files or IPs could still belong to the same malicious campaign.

Table I
ISP NETWORK TRAFFIC STATISTICS.

	$Data_{2011day}$	$Data_{2012day}$	$Data_{2012week}$
# of clients	14,649	18,354	28,285
# of HTTP requests	28,544,473	40,522,026	168,726,091
# of Servers	92,517	117,507	354,578
# of URI Files	1,521,249	2,936,082	12,698,176

IV. EVALUATION DATA

We now describe the details of the network dataset and the ground truth used to evaluate SMASH.

A. Network Trace

To evaluate our system, we experiment with real network traffic traces collected at the edges of a large ISP. We monitored all the incoming and outgoing traffic in the network. The monitored users were mostly residential, and connected to the Internet via high-end ADSL links. Our traces are PCAP files and for every TCP connection and UDP flow, we collected the first 5000 bytes, including the IP addresses and domain names of the destination servers. Table I presents the information of the ISP traffic we collected at different times: one day data from October 2011 ($Data_{2011day}$), one day data from August 2012 ($Data_{2012day}$), and one week data from October 2012 ($Data_{2012week}$). We choose data from different periods to evaluate the performance of SMASH over time.

B. Ground Truth

To estimate the false positives and negatives of our inference results, we use the following data sources as the ground truth.

Intrusion Detection System (IDS): We used a well-known commercial IDS with signatures to label malicious flows with corresponding threat identifiers. Note that since IDS signatures are constantly updated, we used two versions of the same IDS, one from early 2012 and the other from June 2013. We run all the collected network traces through both IDS versions and generate two ground truth datasets: the servers (IDS_{2012}) labeled by the 2012 IDS signatures and the servers (IDS_{2013}) labeled by the 2013 signatures but **not** in IDS_{2012}.

Online Blacklist: We also check our inferred results with popular blacklists, including Malware Domain Block List [4], Malware Domain List [5], Phishtank [6], SpyEye Tracker [8], ZeuS Tracker [14] and online services such as Virustotal [11], Web of Trust (WOT) [12] and WhatIsMyIPAddress [2]. If a server is listed as malicious by any of these blacklists, except WhatISMyIPAddress, we confirm it as a malicious server. As for WhatIsMyIPAddress, which integrates results from 78 blacklist services, we require a malicious report from at least two blacklists to confirm as a malicious server.

Table II
NUMBER OF MALICIOUS CAMPAIGNS.

	$Data_{2011day}$				$Data_{2012day}$			
Infer Thresh.	0.5	**0.8**	1.0	1.5	0.5	**0.8**	1.0	1.5
SMASH	34	**17**	11	6	38	**19**	12	2
IDS_2012_total	1	**1**	1	0	0	**0**	0	0
IDS_2013_total	0	**0**	0	0	0	**0**	0	0
IDS_2012_partial	4	**3**	3	0	0	**0**	0	0
IDS_2013_partial	1	**0**	0	0	2	**0**	0	0
Blacklist_partial	16	**10**	5	6	13	**12**	7	1
Suspicious	4	**0**	0	0	9	**2**	1	0
False Positives	8	**3**	2	0	14	**5**	4	1
FP (Updated)	4	**1**	1	0	7	**1**	1	0

V. EVALUATION RESULTS

A. Inference Results

1) Number of Malicious Campaigns: We first evaluate inference results in terms of malicious campaigns.[7] Table II reports the number of malicious campaigns the SMASH inferred with different $thresh$ we described in Section III-C. For those inferred campaigns, we verify them with our ground truth. If all the servers of a campaign are confirmed by the IDS, we term it as "IDS_2012/2013_total." If only a subset of the servers in a campaign are confirmed by the IDS, we term it as "IDS_2012/2013_partial." If none of the servers of a campaign are confirmed by IDS but confirmed by online blacklist, we term it as "Blacklist." For the campaigns that can not be confirmed by either IDS or Blacklist, we further check the HTTP request status code of those servers from the network traffic, and send the HTTP requests to verify the existence of those servers.[8] If at least half of the servers in a campaign have error code in their network traffic or do not exist any more, we consider this as a "suspicious" campaign. All other campaigns are considered as false positives. Note that there may exist malicious campaigns that are labelled as false positives because we do not have enough information to confirm them. Thus, the false positives here should be an upper bound for our system.

For $Data_{2011day}$ with threshold 0.8, SMASH infers 17 malicious campaigns. Among these, one campaign has all the servers confirmed by 2012 IDS signatures. There are three campaigns where some of their servers are confirmed by 2012 IDS signatures. There are ten campaigns that have their servers partially detected by blacklists. Three campaigns are false positives. If we reduce the threshold to 0.5, SMASH identifies 34 malicious campaigns but the false positives increase to eight. On the other hand, if we increase the threshold to 1.0 and then to 1.5, SMASH detects 11 and 6 campaigns, but with two and zero false positives, respectively. Similar results are observed from $Data_{2012day}$.

[7]Due to the page limit, here we only discuss the campaigns that have at least two involved clients.

[8]We only check the existence of those domains. Our intuition is that malicious domains usually have a short lifetime, and thus might have expired while benign domains usually have a longer lifetime.

Further analyzing the false positives, we discovered two major categories of false positives: Torrent and TeamViewer [10], a remote online collaboration tool. For the Torrent category, several P2P clients connect to a large number of torrent servers by only requesting "scrape.php" files. Thus, they share at least the same filename and sometimes the same IP addresses. For TeamViewer, it has a large pool of servers that are used by their clients to retrieve their ID, which leads to sharing the same path name among those servers. By removing the false positives of these two "noisy" campaigns, we have very few false positives as shown in the last row (FP Updated) of Table II.

2) Number of Servers in Malicious Campaigns: Table III shows the inference results of the number of servers involved in malicious activities. Similar to malicious campaign, if a server is confirmed by the 2012 IDS signatures, we term it as IDS_2012, and if a server is confirmed by the 2013 IDS signatures but **not** by the 2012 IDS signatures, we term it as IDS_2013. For those servers that are not confirmed by either IDS signatures but confirmed by the blacklist, we term it as "Blacklist." All the servers in "suspicious" attack campaigns (as described in Section V-A1) are inferred as "suspicious". For the remaining servers, we compare them with IDS and Blacklist confirmed servers in terms of the requested path, User-Agent, and parameter patterns. Servers confirmed through this way are termed as "New Servers", which are previously undetected servers. All other servers are false positives.

For $Data_{2011day}$ with threshold 0.8, SMASH infers 3,156 servers that are involved in malicious campaigns. Among these servers, only 20 are labeled by IDS signatures and 401 are confirmed by the blacklist. Our system can infer 2,701 more servers which is nearly 7 times the servers detected by IDS and blacklists combined. There are 34 false positives and only 16 after excluding the P2P and TeamViewer cases. We can also see that we generate fewer false positives with higher thresholds. For threshold 1.5, there were no false positives for either $Data_{2011day}$ or $Data_{2012day}$. However, this comes at a price of missing many attack campaigns. We therefore select 0.8 as the threshold, where we detect many attack campaigns while the highest false positive rate is only 0.064%. After removing noise, the largest false positive rate is 0.017%.

We see that SMASH discovers many malicious servers that are not discovered by IDS and blacklist. In Table II with threshold 0.8 for $Data_{2011day}$, 13 clusters are partially detected by IDS or blacklist. Among these clusters, IDS detects only 20 servers and blacklists detect 401 servers. On the other hand, SMASH inferred 2,701 servers that are new, previously unknown malicious servers. Those servers either share the similar pattern with IDS confirmed servers in terms of User-Agent, parameter patterns and URI files, etc or are detected by other researches based on the Google search results. This indicates that about 86.5% of these malicious servers could not be detected by simply relying on IDS or blacklists.

Table III
NUMBER OF SERVERS IN MALICIOUS ACTIVITIES.

	$Data_{2011day}$				$Data_{2012day}$			
Infer Thresh.	0.5	**0.8**	1.0	1.5	0.5	**0.8**	1.0	1.5
SMASH	3,222	**3,156**	3,039	845	407	**287**	150	9
IDS_2012	20	**19**	19	0	0	**0**	0	0
IDS_2013	2	**1**	1	0	3	**0**	0	0
Blacklist	413	**401**	389	74	67	**55**	29	2
New Servers	2,713	**2,701**	2,626	771	171	**152**	91	0
Suspicious	13	**0**	0	0	27	**5**	2	0
False Positives	61	**34**	4	0	139	**75**	28	7
FP (Updated)	22	**16**	2	0	32	**5**	2	0

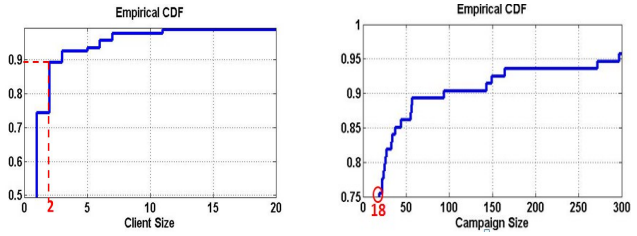


Figure 5. Distribution of the client and campaign sizes.

Furthermore, we see from Table III that without any training or signature updating, SMASH infers malicious severs that are detected by new IDS signatures but missed by old IDS signatures. This shows that SMASH can detect zero-day malicious campaigns before IDS signatures get updated.

Finally we measure the malware servers detected by the IDS but missed by SMASH, i.e., false negatives. To get the ground truth of malware server groups from IDS labels, we group the IDS-labeled malicious servers based on the IDS threat identifier, assuming all the servers in the same threat identifier belong to the same malicious campaign. We have a total of 26 missed malware servers for $Data_{2011day}$ and 27 for $Data_{2012day}$. There are two major types of false negatives. First, there are 40 malicious servers (in the Cycbot, Fake AV and Tidserv threat labels) that do not share any secondary dimension, thus are missed by our system. However, most of those servers share the same URI parameters pattern. Thus, if we extend our URI file dimension to consider the parameter pattern, we could detect these threats. Second, several false negatives are caused by our pruning/filtering process because these servers share the same referrer. SMASH has room for improvement and nevertheless, it can be a great complementary tool to existing approaches.

3) Activity Scale of Malware Campaigns: We measure the scale of malware campaigns by observing the number of clients and servers involved in each malicious campaign. Figure 5 presents the distribution of the campaign size and the client size. We see that about 75% of the attack campaigns have the size smaller than 18, which indicates that most attack campaigns do not connect to a large number

Table IV
ATTACK CATEGORIES.

Activity	Category	# of Servers
Communication	C&C	30
	Web exploit	1
	Phishing	5
	Drop zone	2
	Other malicious servers	1,120
Attacking	Web scanner	23
	iFrame injection	14

Table V
NUMBER OF MALICIOUS CAMPAIGNS DURING $Data_{2012week}$.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
SMASH	31	36	51	40	34	47	51
IDS_2013_total	1	1	1	0	1	1	1
IDS_2013_partial	3	5	7	4	3	8	8
Blacklist	14	19	28	19	16	18	25
Suspicious	4	5	3	4	3	4	6
False Positives	9	6	12	13	11	16	11
FP (Updated)	3	3	11	9	6	12	9

Table VI
NUMBER OF SERVERS INVOLVED IN MALICIOUS ACTIVITIES DURING $Data_{2012week}$.

	Day 1	Day 2	Day 3	Day 4	Day 5	Day 6	Day 7
SMASH	1023	1246	1481	1157	911	1286	1301
IDS_2013	7	15	27	11	7	13	13
Blacklist	371	645	726	348	253	354	698
New Servers	467	398	586	668	443	737	497
Suspicious	13	19	8	18	10	8	21
False Positives	165	169	134	122	198	174	72
FP (Updated)	82	14	130	36	24	89	52

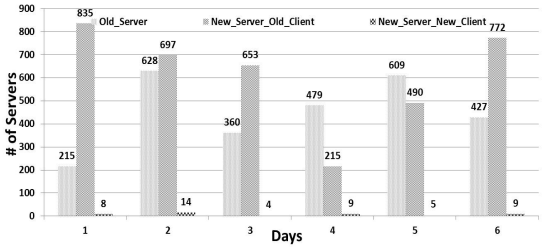


Figure 6. Persistent vs dynamic campaigns.

of malicious servers. However, campaigns with size larger than 18 are usually attacking campaigns, which attack a large number of benign servers (e.g, web scanning and iFrame injection attacks). As for the number of involved clients, 75% of attack campaigns have only one infected client. This result suggests that most client-side clustering systems [21, 22] might be ineffective because they need to correlate among multiple infected clients in the same network.

B. Attack Diversity & Persistency

To demonstrate that SMASH is not limited to only certain types of malicious campaigns, we evaluate SMASH from two different perspectives: attack categories and persistence of servers involved in malicious activities.

SMASH infers diverse malicious campaigns: Our inference results include the attack campaigns related to malicious communication activities and web attacking activities. Table IV categorizes part of our inferred servers involved in malicious campaigns based on IDS labels and Online Blacklists. The servers belonging to communication activities are typically malicious servers, such as those involved in botnet activities and web exploits. The servers belonging to attacking activities are usually benign websites that are targeted by malware, such as web scanning and iFrame injection.

SMASH infers both persistent and agile malicious campaigns: Persistent malicious campaigns are a set of servers that continue to communicate with infected clients for multiple days. On the other hand, agile malicious campaigns are a set of newly identified servers that are communicated by known infected clients. To study the evolution of persistent and agile malicious campaigns, we test our system with one week data from 2012, $Data_{2012week}$. Tables V and VI show the inference results.[9] We consider the first day of the week data as the benchmark. Figure 6 shows the results of each day, where there are 1,014 servers and 27 clients involved in malicious activities at the benchmark day. We see that SMASH infers both persistent (Old_Server in Figure 6) and agile malicious campaigns (New_Server_Old_Client). It can also infer new campaigns (New_Server_New_Client in Figure 6). In addition, we observe that most servers belong to agile malicious campaigns. This result suggests that malware may change their servers/domains every day to evade existing domain-based detection.

C. Effectiveness of the Main and Secondary Dimensions

1) Main Dimension: 24,964 servers in $Data_{2011day}$ and 33,603 servers in $Data_{2012day}$ are dropped after the main dimension processing because they can not be correlated with other servers in client similarity. For those remaining servers, we further investigate the relationship among those servers in the same ASH.[10] To do this, we manually study 50 randomly chosen campaigns from each day. 60% of ASHs are "referrer groups," in which all servers in the same group are referred by the same server. Such groups can be further filtered by the pruning process. 10% of ASHs are "redirection groups," in which all servers in the same groups belong to a redirection chain. Such groups can be also filtered by the pruning process. 8% of ASHs are "similar content groups," in which all servers share very similar content. We further analyze those servers and most of them belong to adult web servers. 18% of ASHs are "unknown

[9]For the campaign with one client, we use threshold 1.0 while for the campaign with more than one client, we use threshold 0.8.

[10]Here, we ignore ASH with only one client, as all the servers in this case are correlated together only because they are visited by one client.

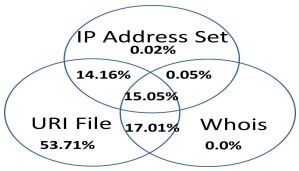


Figure 7. Effectiveness of secondary dimensions.

groups," in which we can not directly find any relationship among those servers. However, they are visited by similar sets of clients. Most of these servers are different companies selling different products or services. The remaining 4% of ASHs belong to malicious ASHs. None of these servers is detected by the IDS while SMASH did.

2) Secondary Dimensions: In SMASH, the ASHs need not satisfy all secondary dimensions. Hence, we measure the effectiveness of each secondary dimension. Figure 7 is the decomposition of inferred servers. We see that 15.05% of the servers satisfy all secondary dimensions and there is no false positive for these herds. URI File dimension is the most effective secondary dimension, which by itself contributes to 53.71% of the detected servers. Although Whois and IP Address Set dimensions individually are not very effective, these two dimensions can help the URI File dimension to confirm more suspicious ASHs. For example, 14.16% are inferred through the combination of IP Address Set and URI File dimensions, and 17.01% are inferred through URI File and Whois dimension combination.

D. Attack Campaign Case Study

We investigate with case studies the advantages of S-MASH over detecting each malicious server in isolation.

1) Capturing the Insight of Malicious Activities: ASHs help us understand the malicious campaign in a holistic fashion. Table VII shows the Bagle botnet [1], which is a mass-mailing computer worm campaign that SMASH inferred. In this campaign, the bot first goes to some servers to download an encrypted file "file.txt" and then connects to C&C servers by requesting "new.php" with the same parameter pattern "p=[]&id=[]&e=[]". There are 94 servers involved in this campaign and they can be clustered in two categories; 40 downloading servers and 54 C&C servers. None of the downloading servers was detected by the IDS or blacklists. Only three C&C servers were detected by VirusTotal. Without the holistic approach of SMASH, we would not have captured the downloading servers and many additional C&C servers.

Table VIII shows a Sality botnet [7] campaign also inferred by SMASH. There are 12 servers involved in this campaign. All have been labeled by the IDS but only eight have been labeled by the blacklists. Again, we cluster the servers in this campaign into C&C servers and downloading servers. Two C&C servers are inferred because they share the same set of IP addresses, the same filename "/" and the same registration information; thus they form a strong ASH. Downloading servers form different ASHs based on the shared filenames. Different from the Bagle botnet, only downloading servers are benign websites compromised by the attackers. Thus, they do not share IP addresses or Whois information. The Sality botnet campaign is inferred by merging these ASHs. The bots might first go to the compromised servers to download additional malware through requesting ".gif" files. They then go to C&C servers to get further instructions.

Based on above two examples, most of downloading servers and some C&C servers from the Bagle botnet are compromised websites. Therefore, domain-reputation-based systems [17] or similarity-based detection systems [16] would not detect such malicious servers.

2) Finding More Malicious Activities: Table IX shows a web injection attack campaign, where infected hosts inject malicious iFrames to benign websites. SMASH inferred 600 benign servers suffering from such attacks while the IDS labeled only four of such attacks, missing more than 99% of the servers. All of these inferred servers are queried by the same set of clients with the same file "sm3.php" under different paths. The servers not labeled by the IDS share the same UserAgent "-" in their HTTP requests to the IDS labeled servers. This confirms that they belong to the same attack campaign. Note that most of the URIs have the path "wp-content," which indicates that those servers are installed with WordPress web application. This attack campaign explores WordPress vulnerability to upload a malicious script "sm3.php."

Table X is the Zeus botnet [13] that SMASH also inferred. This campaign includes eight C&C servers of Zeus. 2012 IDS signatures labeled none of these domains while the blacklists detected only one domain. However, 2013 IDS signatures detected all of these domains. This shows that, as

Table IX
IFRAME INJECTION ATTACK.

Server	URI	UserAgent
smileenh????.co.uk	/images/sm3.php	'-'
dorsets????.org	/images/sm3.php	'-'
calu????.it	/images/sm3.php	'-'
zi??.nl	/wp-content/uploads/sm3.php	'-'
...	...	...

Note: for the privacy protection reason, we use ? to mask part of the detected domains.

Table X
ZEUS BOTNET.

Zeus C&C Server	URI
4k0t155m.cz.cc	/login.php
4k0t177m.cz.cc	/login.php
4k0t144m.cz.cc	/login.php
4k0t166m.cz.cc	/login.php
4k0t111m.cz.cc	/login.php
...	...

Table VII
BAGLE BOTNET.

Categories	Servers	URI	UserAgent	Parameters
C&C Domain	novitacolori.it	/images/news.php	Internet Exploder	p=16435&id=21799517&e=0
	beachrugbyfestival.com	/images/news.php	Internet Exploder	p=16435&id=21799517&e=0
	beautywoman.sk	/images/news.php	Internet Exploder	p=16435&id=21799517&e=0
	...	...	...	...
Downloading	lajuve.org	/images/file.txt	Mozilla/4.0 ...	na
	shayesteganscn.com	/images/file.txt	Mozilla/4.0 ...	na
	www.bigdaybreaker.com	/images/file.txt	Mozilla/4.0 ...	na
	...	...	...	...

Table VIII
SALITY BOTNET.

Categories	Servers	URI	UserAgent	Parameters
C&C Domain	kukutrustnet777.info	/	KUKU v5.05exp =22667130988	22adcdc=72726968
	kjwre9fqwieluoi.info	/	KUKU v5.05exp =22667130988	e65564=135856260
Downloading	merc-connect.com	/images/mainf.gif	KUKU v5.05exp =22667130988	8fff57=84933135
	meta-kit.com	/images/logos.gif	KUKU v5.05exp =22667130988	4f152d=10365530
	fashionenigma.com	/images/logos.gif	KUKU v5.05exp =22667130988	6f2483=58270744
	...	...	...	...

SMASH does not need to update signatures, it can detect zero-day attack campaigns. This campaign seems to be using a DGA-based algorithm to generate similar domain names. All these domains are requested by the same set of clients, and share the same IP addresses and filename "login.php." We searched these domains in Google, and only "4k0t111m.cz.cc" is confirmed as "zeus tracker."

VI. DISCUSSION

Overhead: SMASH is designed to monitor the traffic at the edge of a network. Thus, it can be deployed at enterprise or ISP networks. The most expensive computation part of SMASH is on the similarity calculation, whose complexity is N^2 where N is the number of servers, as we need pairwise similarity among different servers. However, the complexity of similarity calculation can be significantly reduced by using Bloom filters [3] or sparse matrix multiplication [19].

Extensions: In our current implementation, we have three secondary dimensions. Since secondary dimensions are complementary dimensions to further characterize the relationship among servers, they can be extended. For example, we can add time based dimensions [20] to characterize the relationship among servers. We can also add payload similarity to characterize downloading similarity among servers.

Limitations: SMASH assumes that cyber criminals use multiple servers to conduct their malicious activities. Thus, if an attacker uses only a single server to conduct malicious activities (which is now very rare), SMASH can not detect it. In addition, since we use second-level domain for the inference, we might miss the malicious servers using dynamic DNS or hosted on third-party cloud servers. However, those services could incur significant financial cost to the attackers.

Evasions: SMASH relies on the correlation between the main and secondary dimensions. Thus, an attacker who gains the knowledge of SMASH might try to mislead our system by manipulating their relationships as follows:

Evading the Main Dimension: To mislead the main dimension, an attacker can make their bots visit many benign domains with the same URI file.[11] In this case, our main dimension might generate ASHs that include both benign and malicious servers. However, since our client similarity looks at the similarity among all the client sets, it is difficult for an attacker to assure that there are no other benign clients that visit those benign domains. Even when an attacker can use some benign servers to mislead our system, their malicious servers are still included in ASHs that SMASH inferred, which can be further filtered through other heuristics. For example, benign domains might not have such URI files, which may return an error code. In addition, an attacker can also let different bots communicate with different servers to prevent us from generating ASHs. However, this would be a very costly method for attackers, as the more bots they have, the more servers they need to register.

Evading Secondary Dimensions: Compared with the main dimension, secondary dimensions can be relatively easily changed. However, the process is very inconvenient and costly for the attackers. For example, to evade the IP dimension, an attacker can fast flux the IP addresses of their servers, which is very expensive yet easily detected [23]. To evade the URI File dimension, an attacker can assign different names for different servers. However, it makes their connections less scalable; for the attacking campaigns, it usually targets the vulnerabilities of certain files, and thus an attacker can not change such filenames. Although an attacker may successfully evade one of the secondary dimensions, it

[11] There is a very low possibility that the benign domains share similar IP addresses and Whois information with the malicious servers.

is non-trivial to simultaneously evade all dimension to avoid being detected by SMASH.

VII. Conclusion

We studied the malicious servers from a new perspective, i.e., associated server herds. Instead of studying each malicious domain in isolation, we investigated the relationship among servers that are involved in the same malicious activities. This approach enables us to find more malicious servers including servers involved in attacking activities as well as servers communicated with malware. We proposed a novel unsupervised inference system, SMASH, to uncover attack campaigns based on the correlation among associated server herds. Our evaluation with real-world network traces from a large ISP showed that SMASH detects new attack campaigns with a low false positive rate of 0.064% without any training data. Our inferred results also help us capture the insight of the whole attack campaign.

VIII. Acknowledgments

This material is based upon work supported in part by the the National Science Foundation (NSF) under Grant no. CNS-1314823 and CNS-0954096, the Qatar National Research Fund (QNRF) under Grant no. 5-648-2-264, and a Narus Inc. gift award. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of NSF, QNRF and Narus.

References

[1] The bagle botnet. http://securelist.com/analysis/36046/the-bagle-botnet/.

[2] Blacklist check. http://whatismyipaddress.com/blacklist-check.

[3] Bloom filter. http://en.wikipedia.org/wiki/Bloom_filter.

[4] Malware domain blocklist. http://www.malwaredomains.com/.

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

[6] Phishtank. http://www.phishtank.com/.

[7] Sality botnet. http://en.wikipedia.org/wiki/Sality.

[8] Spyeye tracker. https://spyeyetracker.abuse.ch/.

[9] Symantec Internet security threat report. http://www.symantec.com/security_response/publications/threatreport.jsp.

[10] TeamViewer. http://www.teamviewer.com/.

[11] VirusTotal. https://www.virustotal.com/\#url.

[12] Wot (web of trust). http://www.mywot.com/.

[13] Zeus botnet. http://en.wikipedia.org/wiki/Zeus_(Trojan_horse).

[14] Zeus tracker. https://zeustracker.abuse.ch/.

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

[16] M. Antonakakis, R. Perdisci, Y. Nadji, N. Vasiloglou, S. Abu-Nimeh, W. Lee, and D. Dagon. From throw-away traffic to bots: Detecting the rise of DGA-based malware. In *USENIX Security*, 2011.

[17] L. Bilge, E. Kirda, C. Kruegel, and M. Balduzzi. Exposure: Finding malicious domains using passive DNS analysis. In *NDSS*, 2011.

[18] V. D. Blondel, J. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. In *Journal of Statistical Mechanics: Theory and Experiment*, 2008.

[19] Aydın Buluç and John R. Gilbert. Parallel sparse matrix-matrix multiplication and indexing: Implementation and experiments. *SIAM Journal of Scientific Computing (SISC)*, 34(4):170 – 191, 2012.

[20] H. Gao, C. Yegneswaran, Y. Chen, P. Porras, S. Ghosh, J. Jiang, and H. Duan. An empirical reexamination of global DNS behavior. In *sigcomm*, 2013.

[21] G. Gu, R. Perdisci, J. Zhang, and W. Lee. Botminer: Clustering analysis of network traffic for protocol- and structure-independent botnet detection. In *USENIX Security Symposium*, 2008.

[22] Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting botnet command and control channels in network traffic. In *NDSS*, 2008.

[23] C. Hsu, C. Huang, and K. Chen. Fast-flux bot detection in real time. In *RAID*, 2010.

[24] L. Invernizzi, S. Miskovic, R. Torres, S. Saha, S.J. Lee, M. Mellia, C. Kruegel, and G. Vigna. Nazca: Detecting malware distribution in large-scale networks. In *NDSS*, 2014.

[25] M. Konte, N. Feamster, and J. Jung. Dynamics of online scam hosting infrastructure. In *PAM*, 2009.

[26] Z. Li, S. Alrwais, Y. Xie, F. Yu, and X. Wang. Finding the linchpins of the dark web: a study on topologically dedicated hosts on malicious web infrastructures. In *IEEE Symposium on Security and Privacy*, 2013.

[27] T. Nelms, R. Perdisci, and M. Ahamad. Execscent: Mining for new C&C domains in live networks with adaptive control protocol templates. In *USENIX Security Symposium*, 2013.

[28] R. Perdisci, W. Lee, and N. Feamster. Behavioral clustering of HTTP-based malware and signature generation using malicious network traces. In *USENIX NSDI*, 2010.

[29] G. Stringhini, C. Kruegel, and G. Vigna. Shady paths: Leveraging surfing crowds to detect malicious web pages. In *CCS*, 2013.

[30] T. Yen and M. K. Reiter. Traffic aggregation for malware detection. In *DIMVA*, 2008.

[31] J. Zhang and G. Gu. NeighborWatcher: A content-agnostic comment spam inference system. In *NDSS*, 2013.