SBotMiner: Large Scale Search Bot Detection

Fang Yu, Yinglian Xie, and Qifa Ke
Microsoft Research Silicon Valley
Mountain View, CA 94043 USA
fangyu,yxie,qke@microsoft.com

ABSTRACT

In this paper, we study search bot traffic from search engine query logs at a large scale. Although bots that generate search traffic aggressively can be easily detected, a large number of distributed, low rate search bots are difficult to identify and are often associated with malicious attacks. We present SBotMiner, a system for automatically identifying stealthy, low-rate search bot traffic from query logs. Instead of detecting individual bots, our approach captures groups of distributed, coordinated search bots. Using sampled data from two different months, SBotMiner identifies over 123 million bot-related pageviews, accounting for 3.8% of total traffic. Our in-depth analysis shows that a large fraction of the identified bot traffic may be associated with various malicious activities such as phishing attacks or vulnerability exploits. This finding suggests that detecting search bot traffic holds great promise to detect and stop attacks early on.

Categories and Subject Descriptors

H.3.0 [**Information Storage and Retrieval**]: Information Search and Retrieval—*General*; C.2.0 [**Computer Communication Networks**]: General—*security and protection*

General Terms

Algorithms, Security, Measurement

Keywords

Web search, search log analysis, botnet detection, click fraud

1. INTRODUCTION

Web search has been a powerful and indispensable means for people to obtain information today. With an increasing amount of Web information being crawled and indexed by search engines, attackers are also exploiting search as a channel for information collection and malicious attacks. For example, previous studies have reported botnet attacks that search and click advertisements displayed with query results to deplete competitors' advertisement budgets [9].

WSDM'10, February 4–6, 2010, New York City, New York, USA.

In this paper, we study more broadly bot-generated search traffic from search engine query logs. We focus on not just malicious click bot traffic, but any query that was submitted by non-human users. Identifying and filtering such bot traffic is critical to search engine operators for a number of important reasons, including click-fraud detection, page-rank computation, and auto-complete feature training. For example, click-through rates can be used to improve query result rankings [2]. The existence of a large number of bot-generated queries may result in either inflated or deflated query click-through rates, hence may have negative impact on the search result rankings.

More importantly, detecting bot-generated search traffic has profound implications for the ongoing arms race of network security. While many bot queries from individual hosts may be legitimate (e.g., academic crawling of specific Web pages), a significantly fraction of bot search traffic is associated with malicious attacks at different phases. In addition to the well known click-fraud attacks that can be commonly observed in query logs, attackers also use search engines to find Web sites with vulnerabilities, to harvest email addresses for spamming, or to search well-known blacklists. More recently, it is reported that attackers have directed phishing attack victims to search engine results to reduce their infrastructure cost [1]. Although many of these malicious activities do not directly target search engines, identifying the attack-related search traffic is critical to prevent and detect these attacks at their early stages.

Nevertheless, despite a few early efforts towards identifying special classes of click-bot attacks (e.g., [16, 32, 20, 21]), there has been little work on studying bot-generated search traffic, in particular those related with a large class of malicious attacks. A number of challenges make this task difficult. First, the amount of data to process in the query logs is often huge, on the order of terabytes per day. Thus any method that mines the data for identifying bot traffic has to be both efficient and scalable. Furthermore, with many botnet hosts available, attacks are getting increasingly stealthy—with each host submitting only a few queries/clicks—to evade detection. Therefore, search bot detection methods cannot just focus on aggressive patterns, but also need to examine the low rate patterns that are mixed with normal traffic. Third, attackers can constantly craft new attacks to make them appear different and legitimate; thus we cannot use the training-based approaches that derive patterns from historical attacks. Finally, with the lack of ground truth, evaluating detection results is non trivial and requires different methodology and metrics than the detection methods.

We present SBotMiner, a system that automatically identifies bot-generated search traffic from query logs. Our goal is to detect stealthy, low rate bot traffic that cannot be easily identified by common threshold-based methods. To do so, we leverage a key ob-

servation that many bot-generated queries are controlled by a common master host and issued in a coordinated way. Therefore, while individual queries may look indistinguishable from normal user issued queries, they often show common or similar patterns when we view them in aggregate. Instead of reasoning whether an individual search event originates from a bot or not, SBotMiner gathers groups of users who share at least one identical query and click, and examines the aggregated properties of their search activities. By focusing on groups, our approach is robust to noise introduced by sophisticated attackers. It is also easier to verify captured groups, compared to individual users.

We have implemented SBotMiner on a large cluster of 240 machines in Dryad/DryadLINQ [14, 31] and it can scalably process 700 GB data in 2 hours. Using sampled data from query logs collected in two different months, SBotMiner identifies 123 million bot-generated page views, which account for 3.8% of the total sampled query traffic. We further perform a detailed study of the detected bot-traffic. Our key findings include:

- Attackers are leveraging search engines for exploiting vulnerabilities of Web sites. SBotMiner Identifies 88K search-bot groups searching for various PHP scripts and ASP scripts.
- Search bots are spread all over the world. Search bots from different countries display different characteristics. Search bots from countries with high speed Internet access, e.g., Japan, Singapore, US, are more aggressive in submitting queries than those from other locations.
- The results of SBotMiner can be useful for other security applications. For example, SBotMiner identifies a phishing attack that tried to steal a large number of messenger account credentials.

To our knowledge, we are the first to perform a systematic study of a broad class of bot-generated search traffic. Although our results and findings are based on sampled snapshots of query logs, they demonstrate that detecting and analyzing bot-generated search traffic has two advantages: It is not only useful to detect and stop attacks targeting directly at search engines, but is also promising as a general method for identifying a wide class of malicious activities in the Internet.

2. RELATED WORK

Search engines receive a large amount of bot-generated traffic. For example, search engine competitors and third-parties often generate bot-traffic to study the query latency and result qualities [10, 23]. Academic researchers could also use scripts to gather information from search engines (e.g., [3, 27]). These types of bots are not related with attacks. However, identifying and filtering them from the query log is essential for mining data and computing statistics.

Beside these legitimate bots, there also exist a number of malicious search bots, for example, click fraud bots. Click Forensics estimated that the percentage of click fraud reached 12.7% in the second quarter of 2009 [8]. Google also reported a type of click bot attack called ClickBot.A [9, 12] that involves more than 100,000 compromised botnet hosts. Each bot host is very stealthy and conducts low frequency click frauds.

To fight click fraud, several methods have been proposed. Some are based on click through rates and duplicated clicks [16, 32], while others [20, 21] use statistical techniques to identify specific patterns of the click fraud traffic [19]. These approaches are mostly postmortem-based detection. Majumdar et al. proposed a content delivery system to verify broker honesty under standard security assumptions [18].

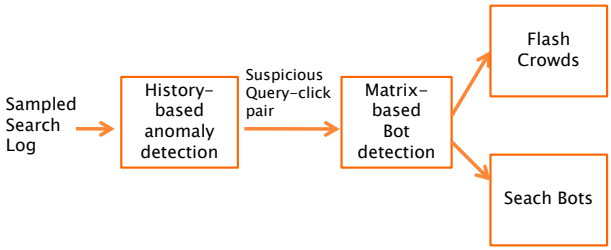


Figure 1: Processing Flow.

Besides click fraud, there are many other malicious activity related searches in the Internet [25]. Moore et al. [22] identified four types of evil searches and showed that some Web sites have been compromised shortly after evil Web searches. For example, attackers searched a known PHP vulnerability that affected all versions of PHPizabi [11]. By searching *"phpizabi v0.848b c1 hfp1"*, attackers can gather all the Web sites that have this vulnerability and then subsequently compromise them.

All the above work detects particular types of search bots based on the prior knowledge of the bot activity characteristics. To our knowledge, there has been no systematic approach to identify search bot in general. A recent study by Buehrer et al. [5] analyzed the search logs and provided high-level characteristics of suspicious search traffic. The study primary focuses on the anomalies in the search logs for detecting aggressive search bots.

However, not all search bots are aggressive. For example, ClickBot.A is stealthy and thus cannot be detected using the common threshold-based methods. Our goal in this paper is to detect these stealthy bot activities. We mainly target distributed search bots that involve many users and span multiple IP addresses, for example, botnet-based search bots.

Botnet detection has received much attention in network security. A large number of study focused on spamming botnets [34, 33, 30, 7, 26, 13]. Recently, botnets are also shown to conduct DDoS attack [24] and steal financial data such as credit card accounts and PayPal accounts [28]. In this paper, we aim to take a first step towards a systematic approach to identify search bots. We compare the captured search bot IP characterizes against the spamming botnet IP properties. We hope our study can shed light on the problem of detecting and understanding search bots in general.

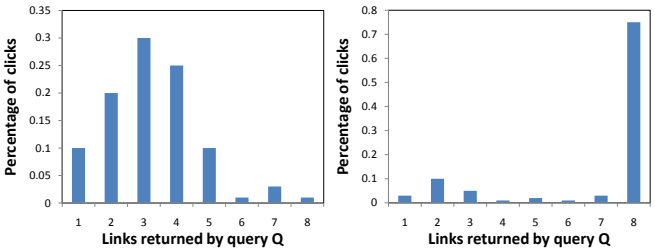
3. METHODOLOGY

SBotMiner explores the distributed nature of stealthy attacks, where distributed bots are controlled by a remote commander. Since bots follow scripts issued by the commander, bot-generated activities are similar in nature. SBotMiner leverages this property and aims to identify groups with similar search activities.

Figure 1 shows the high-level processing flow of SBotMiner. It consists of two steps. The first step is to identify suspicious search activity groups that significantly deviate from history activities. These groups include both the search bots and also flash-crowd events. In the second step, SBotMiner adopts a matrix-based approach to distinguish the search bots from flash crowds. Next, we present these two steps in detail.

3.1 History-based Suspicious Group Detection

The motivations of search bots are usually very different from human users. For example, they could excessively click a link to promote the pagerank, or click competitor's advertisements to de-



(a) (b)

Figure 2: Query-click histogram comparison. (a) history histogram, (b) current histogram.

plete their advertisement budget. In these cases, they want to influence the search engine to change the search results. As a result, query click patterns of search bots must be significantly different from normal users. Also attack traffic is usually short-lived, as many attackers need to rent many compromised machines to perform distributed attacks. Therefore, we use a history-based detection approach to analyze the change of query-click patterns to capture the new attacks.

Each query q to the search engine retrieves a query result page. On the page, there could be many links $\{L_1, L_2, ..., L_n\}$, including all the search results and advertisement links. A user can click zero or more of such links. Note that no click is treated as a special type of click. Given the click statistics, we can build a histogram of click distributions of query q as illustrated in Figure 2. Each bar corresponds to one link L_i and the bar value is the corresponding percentage of clicks. The histogram of query-click distribution may change over time. For example, Figure 2(a) shows a the histogram of query q in history and (b) shows the current histogram.

We use a smoothed Kullback-Leibler divergence to compare the histogram of query click activities against history. Denote $H_s(i), i = \{L_1, ..., L_n\}$, the history histogram of clicks on L_i, and $H_c(i)$ the current histogram of clicks. The histogram $H(i)$ is normalized such that $\sum_i H(i) = 1$. The Kullback-Leibler divergence [17] from current histogram H_c to the history histogram H_s is defined as:

$$D_{KL}(H_c \| H_s) = \sum_i H_c(i) \log \frac{H_c(i)}{H_s(i)} \quad (1)$$

Note that the Kullback-Leibler divergence is always non-negative, i.e., $D_{KL}(H_c \| H_s) \geq 0$, with $D_{KL}(H_c \| H_s) = 0$ if and only if $H_c = H_s$.

Intuitively, D_{KL} measures the difference between two normalized histograms. For each link L_i associated with a given query, the ratio $\frac{H_c(i)}{H_s(i)}$ measures the frequency change of clicks on the link L_i. The log of this ratio is then weighted by the current click frequency $H_c(i)$, and the Kullback-Leibler distance in Equation 1 is the sum of the weighted log ratios over all clicked links $i = \{L_1, ..., L_n\}$.

Kullback-Leibler distance and its variants have been used in many applications including query expansion [6] and text categorization [4] in information retrieval. For our search bot detection application, we make two changes to Equation 1:

- First, we want to detect search bots that are currently active. In other words, we are only interested in links that receive more clicks. To account for this we replace the log ratio by $\max\{\log \frac{H_c(i)}{H_s(i)}, 0\}$.
- Second, if the current click histogram is similar to history histogram but the total number of clicks associated with a query is increased significantly, we still want to mark such query-clicks as suspicious for further examination using a matrix-based method (see next section), especially for those rare query terms that are associated with only one type of clicks (normalized histogram would be the same for these queries). For this purpose, we add the second term $\log \frac{N_c}{N_s+1}$ to the Kullback-Leibler distance, where N_c is the total number of clicks currently received for a given query, and N_s is the total number of clicks associated with the same query in the history (we use $N_s + 1$ to avoid potential overflow when $N_s = 0$.).

The final modified Kullback-Leibler distance becomes:

$$D_{KLm}(H_c \| H_s) = \log \frac{N_c}{N_s + 1} + \sum_i H_c(i) \max\{\log \frac{H_c(i)}{H_s(i)}, 0\} \quad (2)$$

We use a smoothing version of $H_s(i)$ to avoid overflow in D_{KLm}, by replacing zero values in $H_s(i)$ with a small value ϵ. Specifically, we define

$$H_s(i) = \begin{cases} \beta H_s(i), & \text{if } H_s(i) > 0 \\ \epsilon, & \text{otherwise} \end{cases}$$

Here β is a normalization factor such that the resulted histogram satisfies $\sum_i H_s(i) = 1$.

If there is significant difference ($D_{KLm}(H_c \| H_s) > \alpha$) between the history histogram and current one, we conclude the related query q as a suspicious query. For a link that is more popular than history, we pick it as the suspicious click c and form a query click pair <q,c> (abbreviated to QC pair in the rest of the paper, and the lowercase qc denotes a particular QC pair value). Note that for one query, if multiple links are becoming popular, multiple QC pairs are generated. In our experiment, SBotMiner conservatively sets α to be 1 and also requires each group to have at least 100 users so that we can study the group similarity of these users.

Our history-based detection captures events that suddenly get popular. These events can include both bot events and flash-crowd events. Next, we use a matrix-based approach to distinguish the two cases.

3.2 Matrix-based Search-bot Detection

The main difference between the bot traffic and flash crowds is that bot traffic is generated by a script. In contrast, flash crowd activities are originated from human users. Therefore, the flash crowd groups exhibit a higher degree of diversity. In other words, although the users who generate the flash crowd traffic share the same interest at one time, e.g., searching Michael Jackson and clicking his webpage, they can have different search history and also diverse system configurations such as different browsers and operating systems. In this paper, we use the query history as a feature to drive the matrix-based approach, and use other features such as system configurations for validation (Section 4.2).

The matrix-based algorithm leverages the diversity of users to distinguish bot events from flash crowds. For each suspicious QC pair qc detected by the history-based algorithm, we first select all users U_{qc}= $\{U_1, U_2, ..., U_m\}$ who performed this query-click activity and then extract all the search traffic from U_{qc} into a group G. Suppose there are n unique queries $\{Q_1, Q_2, ..., Q_n\}$ in the group, we construct a matrix M_{qc} as shown in Figure 3. Each row in the matrix corresponds to a query and each column represents a user. $M_{qc}(i, j)$ denotes the number of query Q_i originated from user U_j.

Figure 4 illustrates two representative matrixes. Figure 4(a) is a flash-crowd matrix, where users share one identical query q (last

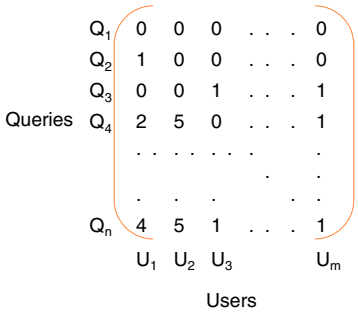


Figure 3: Matrix Representation.

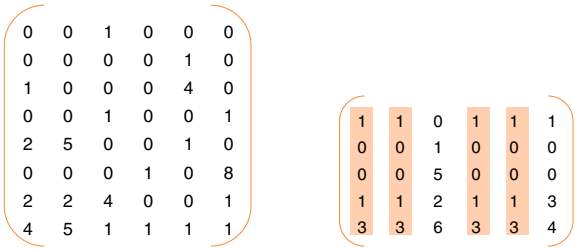


Figure 4: Query-click histogram comparison. (a) a flash-crowd matrix, (b) a bot matrix, shaded columns are correlated users.

row), but their other queries are diverse and uncorrelated. Figure 4(b) is a bot matrix, where many users (shaded columns) in this matrix have identical/correlated behavior.

In some extreme cases, bot activities are very dedicated, with each bot user only issuing one or a few queries. For these easy cases, we use a metric F_{qc} to quantify the "focus-ness" of the group, i.e., percentage of traffic originating from users that searched only for q over the total traffic in G.

$$F_{qc} = \frac{\sum_j \{M_{qc}(q,j) | \forall j, s.t. \sum_{i \neq q} M_{qc}(i,j) = 0\}}{\sum_{ij} M_{qc}(i,j)} \quad (3)$$

Figure 5 shows the distribution of F_{qc} across all groups that are detected by the history-based scheme. We can see that more than 10% of groups have F_{qc} equal to zero. This shows that users who perform these qc pairs all conduct some other queries as well, suggesting that these groups to be flash crowd groups. There is a small fraction of groups with F_{qc} between 0.1 and 0.6. A majority (70%) of the groups have F_{qc}>0.7. For these groups, at least 70% of users conduct the qc pair query do not issue other queries. When F_{qc} is close to 1, it means almost all the users in the group search only q. They are very suspicious groups as normal users have diverse activities. SBotMiner conservatively sets F_{qc} = 0.9 as the threshold for selecting bot groups.

By looking at the user-group properties, our approach is robust to a small amount of noise (i.e., coincident legitimate search traffic). It is also easier to validate bot-groups than individual bots because we can compare the similarity of the activities within a group, e.g., whether they use the same user agent.

3.3 Separating Bot Traffic from User Traffic

The matrix-based detection method presented in the last subsection provides a base for us to detect groups with a dominant fraction of bot search traffic. However, using a single fixed threshold

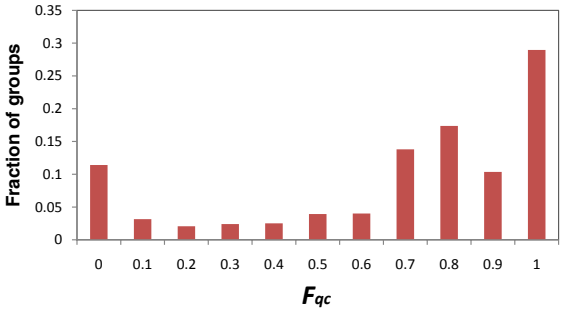


Figure 5: Fraction of groups vs. F_{qc}.

of the group "focus-ness" F_{qc} requires us to set a very conservative threshold (close to 1) to reduce the false positives. If an attacker picks a more popular query by normal users, the fraction of bot traffic may not be large enough to meet the strict threshold and hence SBotMiner may miss the corresponding group in detection. Furthermore, each bot user may submit multiple queries, and therefore appears to be normal users. In these two cases, we would still like to catch the bot-traffic group and further separate bot-generated query/clicks from normal user traffic.

To do so, we perform principal component analysis (PCA) [15] on those query matrices that do not meet the "focus-ness" threshold. PCA is a statistical method to reduce data dimensionality without much loss of information. Given a set of high-dimensional data points, PCA finds a set of orthogonal vectors, called principal components that account for the variance of the input data. Dimensionality reduction is achieved by projecting the original high-dimensional data onto the low-dimensional subspace spanned by these orthogonal vectors.

The rational of adopting PCA for our analysis is that since bot users from one group are controlled by the same script, their queries are often strongly correlated, for example, all submitting a similar set of queries. We can therefore use PCA to identify the correlated bot query patterns that are hidden among a large variety of normal user queries.

Given a query matrix M_{qc}, we first convert it into a binary matrix B_{qc}, where $B_{qc}(i,j) = 1$ iff $M_{qc}(i,j) > 0$. PCA is then performed on the converted binary matrix. Since bot user queries are strongly correlated while normal user queries are not, intuitively the subspace defined by the largest principal component corresponds to the subspace of bot user queries if they exist. So SBotMiner selects the largest principal component denoted as E_1 and computes the percentage of data variance P_1 accounted for by E_1. A large P_1 means a large percentage of users all exhibited strong correlations in their query patterns and are thus suspicious.

To further identify these suspicious users, for any matrix B_{qc} with P_1 greater than a threshold (currently set to 60%), the next question is how to identify the column vectors that correspond the subspace defined by E_1. To do so, we project B_{qc} onto the subspace defined by E_1 to obtain a projected matrix B'_{qc}. For each column (user) vector u_i in the original matrix B_{qc}, denote its corresponding vector in the projected matrix B'_{qc} as u'_i. If the L_2 norm difference between u_i and u'_i is very small, that means the projection onto the subspace B'_{qc} does not change the vector u_i much and the principal component E_1 describes the data well. Therefore, SBotMiner selects the k user vectors with the smallest L_2 norm differences $||u_i - u'_i||$ that accounts for the energy in the subspace of E_1. To do so, we choose k such that

$$k = \lfloor m \times \frac{||E_1||_2}{||E||_2} \rfloor$$

	Features
For detection	User ID (anonymized)
	Query term
	Click
For validation	IP address
	Cookie (anonymized)
	Cookie creation time
	User agent (anonymized)
	Form
	Is Javascript enabled

Table 1: Features.

Time	Total Sampled Pageviews
Feb data	1,722,390,355
April data	1,662,815,486

Table 2: Data.

The corresponding k users are the suspicious users in a group as their query vectors changed the least from the original space after projecting into the reduced 1-dimension subspace.

In practice, since the number of columns (users) m in a matrix can be very large, to reduce the computation complexity, we sample 1000 users to construct a smaller sampled query matrix M_{qc} and perform the above computation on only the sampled matrix. Correspondingly, the k selected suspicious users are only from the sampled matrix. In order to identify all such similar suspicious users, we look back into the query log and identify all the users that have identical search patterns (i.e., with identical query/click pairs) to at least one of the suspicious users from the sample matrix. The output are therefore the expanded bot-groups.

4. RESULTS AND VALIDATION

We sampled two months of search logs (February and April 2009). Each month of sampled data contains over 1 billion pageviews as shown in Table 2. A pageview records all the activities related to a search result page. From a pageview, we extract nine features as shown in Table 1: user ID(recorded in the cookie), query terms, clicks, query IP address, cookie, cookie creation date, user agents, FORM, and whether Javascript is enabled. Due to privacy concerns, user ID, cookie, and user agents are anonymized by hashing. In addition, when we look at IP addresses in the log, we focus mostly on studying the IP addresses in a region, rather than examining a particular IP address.

Among these features, the first three features (user ID, query terms, and clicks) are used for detection and the remaining six are used for validation.

We implement SBotMiner on the Dryad/DryadLINQ platform, where data are processed in parallel on a cluster of 240 machines. We first partition the data according to queries and examine the query-click histograms of different queries in parallel. The output of this step is a set of suspicious QC pairs. In the second step, we partition the data according to users and process users in parallel. This step extracts all the activities of users that are associated with suspicious QC pairs. Finally, each QC pair is processed in parallel using the matrix-based detection scheme. The entire process of SBotMiner can be finished within 2 hours to process the 700GB of sampled data.

4.1 Detection Results

Table 3 summarizes our detection results. The history-based detection scheme detects more than 500K groups in February and

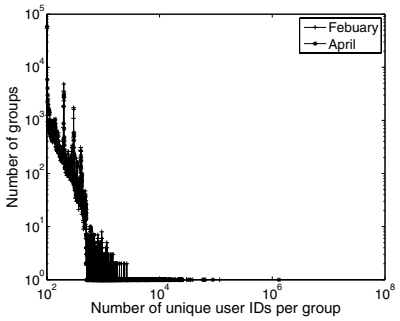


Figure 6: Number of unique users per search bot.

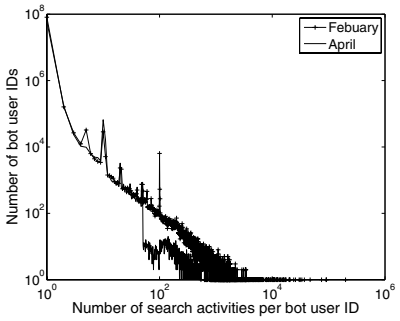


Figure 7: Number of search activities per user.

480K groups in April. Among these groups, a dominant fraction of the bot-search activities can be detected using the simple group "focus-ness" F_{qc} metric. With $F_{qc} = 0.9$, the matrix-based approach identifies 280K groups in February and 322K groups in April to be bot groups. In total, these groups involve over 60 million pageviews in February and 63 million pageviews in April, which roughly account for 3.8% of the total sampled traffic.

We look at the number of user IDs per searchbot group detected with the F_{qc} metric. Figure 6 shows the distribution. The x-axis is the number of unique user IDs per group and y-axis is the corresponding number of groups. We can see that over 58K groups are small (with just 100 user IDs, where 100 is our threshold of the distributed search bot). There are several large bot groups, with the largest one containing 1.3 million user IDs.

Figure 7 shows the search activity per user ID. A majority of the user IDs are quite transient, conducting only one search. This could also be caused by the fact that some search bots disable cookies, so each search query will generate a new user ID. However, we do capture some bot users that are aggressive. The most aggressive user conducted 199K searches. This user may have utilized a script to obtain realtime stock price. The script generateed multiple queries per second (see Section 5.3.4 for more details).

For the remaining suspicious groups, SBotMiner performs PCA and detects 137 groups as bot-search groups using both February and April's sample data. Figure 8 shows the cumulative distribution of these group sizes. Majority of the groups (90%) have fewer than 4,000 unique bot-users, with a few groups being very large with more than 10,000 users.

Although the number of groups detected using PCA is small, interestingly, we find a phishing attack among them by analyzing the groups with the most number of users. This attack happened in April, using which data, SBotMiner identifies five large groups involving more than thousands of unique users that all searched the

	Suspicious groups	Bot groups	Total Pageviews	Unique user IDs	Unique IP addresses
Feb data	543,600	280,767	60,335,661	81,511,784	212,109
April data	480,491	322,476	63,525,084	61,342,906	314,341

Table 3: Result Summary

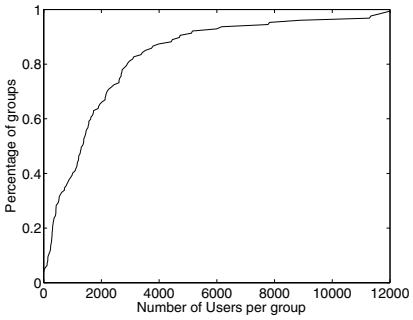


Figure 8: CDF of the number of unique users per group.

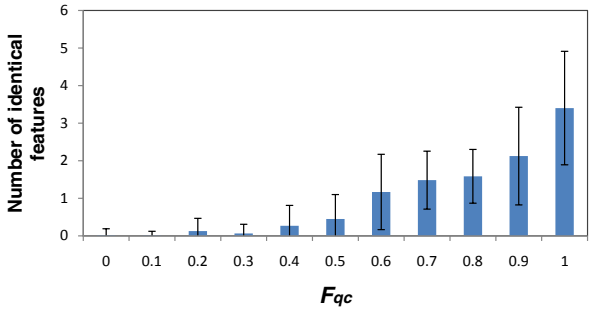


Figure 9: The number of identical features vs. F_{qc}. Each bar plots the average number of identical features per group, error bars represent (+/-) standard deviation.

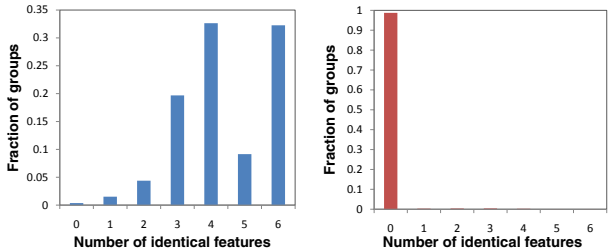


Figure 10: Number of identical features in each group. (a) bot groups ($F_{qc} > 0.9$), (b) flash-crowd groups ($F_{qc} < 0.1$).

keyword "party" for image results. Examining closely, we find that all the users had the following similar format in their referrer field, with a website followed by some user IDs:

http://<domain-name>.com/?userid

Further study suggests these five groups were part of a phishing attack that steals the credentials (login names and passwords) of the users who access social network sites or use messenger services.When this attack compromises a user account through phishing, it logs in as the user to the corresponding service and sends both a message and a URL link to the friends in the contact list. Each such message tells the friend that the user has found a picture of the corresponding friend and the picture is stored in the attached URL link. When the friend clicks the link, she will be directed to a Web site, asking her to log in in order to see the picture. This Web site is set up by the attackers. Once the friend enters her login name and password, she will be redirected to the Live search site using keyword "party" to display a set of image results from the query. Meanwhile, the attack successfully collects another victim's credential and can further propagate the attack by infecting more users. The user IDs attached at the end of the referrer field shows the victim user IDs.

Using this information, we examine all queries that share the similar structure in their referrer fields and identify many more user IDs that are likely the phishing attack victims. For this particular attack, the search traffic is actually not generated from bots, but from a group of real victim users. Although SBotMiner is not strictly detecting bot-traffic in this case, the ability to identify unusual search traffic group is important to detect and stop other attacks as early as possible.

4.2 Validation

We verify the captured bot activities by examining the similarity within the group. As we mentioned in Section 3.2, SBotMiner uses three features for detection (user ID, query and click). Now, we use the remaining features for validation: hash of cookie, cookie creation date, IP address, FORM, hash of user agents. If one group has over 99% of pageviews agreeing on one feature, e.g., sharing the same FORM, we consider that group has one identical feature. If a group agrees on one or more features, it indicates that this group is highly likely to be controlled by one commander, and thus likely to be bot-groups.

Figure 9 plots the number of identical features that each group agrees on when we vary the group focusness threshold F_{qc}. As we can see if a group is very focused, i.e., most users all query one particular query, these users tend to have more than one identical feature. This validates that these groups are very likely to be controlled by one commander. For a group with a low focusness, i.e., users have diverse query terms, users within this group tend to have zero identical features.

As SBotMiner conservatively sets F_{qc} = 0.9 as the threshold for selecting bot groups, Figure 10(a) shows the number of identical features within the selected bot groups. We can see that for most groups, records within a group share three or more features, suggesting they are highly likely to be controlled by one entity (attacker). Only around 0.3% of the groups do not agree on any feature. They may be the false positives of our approach.

As a comparison, we check the similarity of features of flash-crowd like groups. The flash-crowd groups are categorized by those with a sudden increase in volume (captured by history-based detection), but have a very low F_{qc} ($F_{qc} < 0.1$[1]). There are a total of 125,357 flash-crowd groups. Figure 10(b) plots the number of identical features within flash-crowd groups. We can see that over 98.6% of groups do not agree on any feature, showing that they are from many users and are unlikely to be controlled by a single attacker. The dramatic difference between the captured bot groups

[1]We pick 0.1 as threshold because the fraction of groups between 0.1 and 0.6 is very small as shown in Figure 5.