# NEIGHBORWATCHER: A Content-Agnostic Comment Spam Inference System

Jialong Zhang and Guofei Gu
SUCCESS Lab, Department of Computer Science & Engineering
Texas A&M University, College Station, TX
{jialong,guofei}@cse.tamu.edu

## Abstract

*Comment spam has become a popular means for spammers to attract direct visits to target websites, or to manipulate search ranks of the target websites. Through posting* a small number of *spam messages on each victim website (e.g., normal websites such as forums, wikis, guestbooks, and blogs, which we term as* spam harbors *in this paper) but spamming on* a large variety of *harbors, spammers can not only directly inherit some reputations from these harbors but also avoid content-based detection systems deployed on these harbors. To find such qualified harbors, spammers always have their own preferred ways based on their available resources and the cost (e.g., easiness of automatic posting, chances of content sanitization on the website). As a result, they will generate their own relatively stable set of harbors proved to be easy and friendly to post their spam, which we refer to as their* spamming infrastructure. *Our measurement also shows that for different spammers, their spamming infrastructures are typically different, although sometimes with some overlap.*

*This paper presents* NEIGHBORWATCHER*, a comment spam inference system that exploits spammers' spamming infrastructure information to infer comment spam. At its core,* NEIGHBORWATCHER *runs a graph-based algorithm to characterize the spamming neighbor relationship, and reports a spam link when the same link also appears in the harbor's clique neighbors. Starting from a small seed set of known spam links, our system inferred roughly 91,636 comment spam, and 16,694 spam harbors that are frequently utilized by comment spammers. Furthermore, our evaluation on real-world data shows that* NEIGHBORWATCHER *can keep inferring new comment spam and finding new spam harbors every day.*

## 1 Introduction

Spamdexing (also known as web spam, or search engine spam) [25] refers to the practice of artificially improving the search rank of a target website than it should have. The rise of such spam causes unnecessary work for search engine crawlers, annoys search engine users with poor search results, and even often leads to phishing websites or malware drive-by downloads. In a recent study [15], Google reports about 95,000 new malicious websites every day, which results in 12-14 million daily search-related warnings and 300,000 download alerts.

To boost the ranking of the target websites, spammers have already developed lots of spamdexing techniques [25] in the past few years, most of which also called Black Hat SEO (Search Engine Optimization). Text and Link manipulations are two main SEO techniques frequently abused by spammers to exploit the incorrect application of page rank heuristics. Through injecting excessively repeating contents in their websites or changing the perceived structure of webgraph, spammers have successfully improved their search ranks in the past [6]. However, since search engines have already developed new techniques to detect these content/link manipulation tricks [9, 11], spammers begin to change to another new trick of spamdexing, named comment spamming. Comment spamming refers to the behavior of automatically and massively posting random comments or specific messages (with links to some promoting websites) to a benign third-party website that allows user generated content, such as forums (including discussion boards), blogs, and guestbooks. In this paper, we refer to those benign victim websites that are frequently used by spammers to post comment spam as *spam harbors* (or sometimes *harbors* for short). This new trick of comment spam can benefit spammers in several ways: (1) spammers can easily inherit some reputations of these harbors with nearly zero cost; (2) the risk of being detected by search engines is reduced; (3) spammers can easily disguise themselves as normal visitors who also contribute content.

Most existing state-of-the-art approaches to detect comment spam use three types of features: (1) content-based features [24, 31], e.g., the utilization of certain words, content redundancy/frequency, topic or language inconsistency; (2) context-based features [32], which mainly refer to the

existence of URL cloaking or redirection techniques (because normal URLs rarely use them); (3) behavior-based features [33, 23], which mainly refer to the time difference between main article and comment posting (which is typically short in normal cases). Unfortunately all of them have their clear limitations and spammers already become more agile in evading them. To evade content-based detection [24, 31], spammers can artificially manipulate their posting content by mixing spam links with normal content, or posting very few content on each harbor site but spamming on a large variety of harbors. Thus they can easily disguise themselves as normal visitors, and remain not detected. In [32], the authors used context-based features, i.e., checking the use of URL cloaking or redirection tricks, which is effective for certain type of spam. However, it has a low coverage in the detection scope, because most comment spam currently is mainly used for search rank manipulation [33]. Thus, URL hiding/cloaking is no longer necessary and less used. The limitation of time differences between main article and comment posting [33, 23] is also clear; it is applicable to only some blogs, which are time sensitive, while not easily applicable to more broader websites such as forums, guestbooks. As we can see, within these three types of features, the later two typically may not work well alone, thus they are suggested to combine with content-based features. Finally, we note that another limitation for content-based detection is that the training overhead is typically high; it needs to be trained and updated constantly (with the fast-changing web contents) and it has to be customized specifically for each *individual* website.

Complementary to previous work on comment spam detection, we approach the problem from a new perspective, i.e., we focus on exploiting the structure of spamming infrastructure, an essential component of comment spamming (which is relatively stable), rather than the content (which changes frequently). The intuition behinds structure-based inference is that, while spamming content can be dynamic, spamming campaigns and spamming structure are much more persistent. If we can recognize spamming patterns that characterize the structure of spammers' infrastructure, then we can continue to detect spam even if the spammers frequently change their spam content.

Driven by profits, spammers always want to take full advantage of their resources (e.g., spam harbors). Usually there are two ways to achieve this goal: massive spamming on a few harbors, or posting few on each harbor but spamming on a large variety of harbors. If spammers post similar spam content massively on a few harbors, it is easy to be detected by content-based detection systems. However, if spammers post spam on a large number of harbors, the chance of detection at individual harbor is reduced. In particular, spammers typically keep using their familiar harbors, because these websites may not have effective sanitization/filter mechanisms or posting on them can be easily automated, thus making a comfort zone for the spammers.

In this paper, starting from a seed set of collected spam, we first perform a measurement study on spam harbors' quality and the graph structure of spam harbors, which reveals the structure of spammers' infrastructure. We find that spam harbors usually have relatively low qualities/reputations, which is quite counterintuitive because spammers are expected to spam on high quality harbors to maximize their profits. To compensate the low reputations of these harbors, spammers intend to keep using a large variety of harbors for spamming. As for the structure of their spamming infrastructure, we find that spam harbors in the same campaign always post similar spam links at the similar time, which reflects that spam harbors in the same campaign always have close relationships while normal websites do not necessary have such relationships.

Based on observations from this measurement study, we design a system named NEIGHBORWATCHER. Our intuition is that if the promoting link in a comment also appears in the neighbors (cliques in the spam harbor infrastructure) of this harbor, it has a higher possibility of being a spam message, because normal links are not necessary *always* been posted on *the specific set* of harbors that have been verified to be exploited by the same spammer/campaign. NEIGHBORWATCHER uses a graph-based propagation algorithm to characterize neighborhood relationships among collected spam harbors in order to infer the spamming infrastructure. When a new comment is posted with some link, NEIGHBORWATCHER performs a *neighborhood watch* on the graph and calculates a suspicious score based on the graph-based inference. With a prototype implementation running on a real-world dataset, we show that NEIGHBORWATCHER can keep finding new spam and spam harbors everyday.

The major contributions of this paper are summarized as follows:

- We present the first in-depth measurement study of the quality and structure analysis of spam harbors. We observe that most of spam harbors have relatively low reputations, which compels spammers to spam on a large set of harbors. We also find that spammers always have a stable set of harbors and keep using them to post spam unless they are blocked.
- We propose a graph-based propagation algorithm to characterize spamming infrastructure and an inference algorithm to infer comment spam. Our new approach is content-agnostic and does not need training/customization on individual websites.
- We implement a prototype system and evaluate it on real-world data. Through our evaluation, NEIGHBORWATCHER has detected 91,636 spam links and 16,694

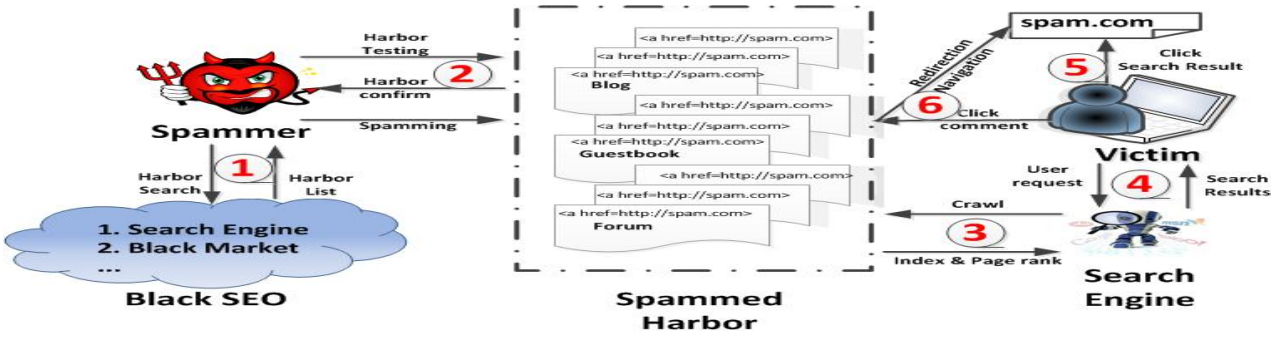



**Figure 1. The Workflow of Comment Spamming**

spam harbors in total. Among these, 1,364 spam URLs have not been indexed by Google yet. In addition, 147 spammer IPs and 4,008 spammer email addresses can not be found in existing spam blacklists. This implies that our system can keep finding new spam, and giving *earlier* warning than Google and existing spam blacklist services.

The rest of the paper is structured as follows. We describe the background and our target problem in Section 2. We discuss the measurement study in Section 3. We describe the design and implementation of NEIGHBORWATCHER in Section 4 and evaluation results in Section 5. In Section 6, we discuss various extensions and limitations of NEIGHBORWATCHER. We present related work in Section 7 and conclude our work in Section 8.

## 2 BackGround

In this section, we provide a brief overview of how existing comment spamming works. Then we present several typical categories of harbors that are frequently used by comment spamming.

### 2.1 Threat Model

As illustrated in Figure 1, in comment spamming, spammers typical need to first find out suitable harbors, e.g., those with good reputations, those that can be automatically spammed, or those that have weak or no sanitization mechanisms. To achieve these goals, spammers usually use different Google dorks [2] to find target harbors or simply buy some harbors from underground markets. In this way, they can get a set of harbors that can be used to automatically post spam (labeled as ①). These collected harbors are usually some forums, blogs, or guestbooks that support user contributed content, and normal users typically contribute a lot on them. In this case, spammers can easily disguise them as normal visitors, which makes content-based detection inefficient. Then spammers need to verify these collected harbors to assure that they can automatically post spam on these harbors without being easily blocked. They can do this by posting random normal content. After verification, spammers begin to spam on validated harbors in a large scale (②). As a result, when search engine bots crawl these harbors, they will index the spam URLs posted in these harbors, which can finally improve the search rank of the promoted websites in the spam (③). Thus, when users search certain keywords through search engines, those promoted spam websites may have higher search ranks than they should have (④), which may lead victims to click spam websites in search results (⑤). Or victims may directly click the embedded spam links when they read those spam comments, which will directly lead them to spam websites (⑥)

### 2.2 Categories of Spam Harbors

As described in Section 2.1, to launch efficient comment spam, spammers need to carefully choose their spam harbors. Next, we describe three most common types of harbors frequently used by comment spamming.

**Web blogs** are typically websites where people can post articles. Usually these blogs have comment areas for each article, which allow visitors to comment on corresponding articles. Thus, spammers can also use these comment space for spamming. A possible detection feature of such spamming is that spammers may post repeating spam comments, and also the time difference between the article and the spam comment could be longer than the normal case [33].

**Forums** are typically websites where people can have conversations in the form of posting messages. Since most forums need users to register beforehand, spammers can also post their spam as long as they can automatically register accounts for these websites. Since some of conversations could last for a long time, it is hard to detect this kind of spam based on the posting time.

**GuestBooks** are typically platforms to let visitors to communicate with corresponding websites. Most companies websites have their guestbook pages to let people leave a message to their companies. GuestBooks are more agile than blogs and forums, because there are no normal timing patterns and no conversations; everyone can leave any message anytime with fewer restrictions.

Table 1 summarizes the effectiveness of existing different types of detection features on different types of harbors. We can see that content-based and behavior-based features are relatively effective for blog spam. Articles in blogs usually have specific topics, thus it is easy to detect spam whose content is not related to articles. Also it is less common for normal users to comment on an out-of-date articles in a blog. Unfortunately, most blogs still do not have any such detection system, which leaves them still to be a good platform for spammers. In the contrast, there are typically no specific topics in guestbooks; everyone can leave any message at any time. Thus guestbook spam is hard for all existing detection features. Context-based features, i.e., detecting URL cloaking, do not have good results on all the harbors because of the very limited effectiveness scope. In short, as we can clearly see that existing detection features are certainly not enough in fighting comment spam. The need for effective and complementary techniques is pressing.

**Table 1. Effectiveness of Different Types of Detection Features**

| Features | Content | Behavior | Context |
|---|---|---|---|
| Blogs | good | good | bad |
| Forums | medium | medium | bad |
| GuestBooks | bad | bad | bad |

# 3 Spam Harbor Measurement

Comment spamming, as a relatively new trick of spamdexing, has been reported for quite a while, ever since 2004 [4]. However, until recently, comment spamming has not been sufficiently studied, and existing approaches are clearly insufficient as discussed earlier. To gain an in-depth understanding of the comment spamming, we study the problem from a new perspective, i.e., spam harbors, which form the basic infrastructure for spammers. Why spammers choose these harbors? Are there any special properties of these harbors? Can we use these properties to help defend against comment spam? In this section, we will try to answer these questions and present what we have learned from these harbors.

## 3.1 Dataset

To collect spam harbors, we started from 10,000 verified spam links $S_{study}$ (which are collected from our previous work [27]) and collected a dataset containing 38,913 real-world spam harbors, which are represented with unique domain names. Specifically, we searched all these spam links in Google and collected the search results. Among these search results, not all of them are spam harbors, e.g., some are security websites that report those search links as spam, and some are benign websites[1] that link to those search links. However, we observe that spam harbors typically contain embedded hyperlink tags (e.g., anchor tag $< a\ href = "..." >$ and BBCode $[URL]...[/URL]$). This is because spammers perform automated posting on massive websites, and typically they are unsure whether their target spam harbors support embedded links or not. Thus, in order to achieve a high success rate of posting the spam links, they choose to use embedded hyperlink tags [34]. Based on this observation, we only extracted those search results with embedded hyperlink tags in their contents as spam harbors.[2]

For each spam harbor, starting from the page that contains our verified spam links, we further crawled all possible pages on that website and recorded timestamps on the page (typically these webpages always record the time when a message is posted). Table 2 provides an overview of our collected data. To study the differences among three categories of harbors, we roughly group the collected harbors based on their URLs. That is, if a harbor URL contains keywords such as "blog","forum","guestbook", we will cluster them in blog, forum, and guestbook category, respectively. We do not further distinguish remaining harbors without clear keywords (listed as "other" in Table 2), and treat them as the mixing of these three categories. Among 38,913 spam harbors returned by search results, 35,931 are still active so we can crawl further pages on these harbors. We have crawled more than 9 million postings in total.

## 3.2 Quality of Harbors

Since the goal of comment spamming is to improve the search ranks of spam websites, the higher quality spam harbors have, the more effective comment spamming is. In this section, we try to evaluate the quality (e.g., reputation) of these spam harbors in the following three perspectives.

**PageRank Score** is calculated by PageRank algorithm [29], which is widely used by search engines to rank the

[1] Since some of those search links are compromised benign links, they may also be linked by other benign websites.

[2] Note that we may not extract a complete list of harbors in this way. Instead, our conservative goal here is to extract spam harbors with a higher confidence.

**Table 2. Data Collection of Comment Spam Harbors**

| | blog | forum | guestbook | other | total |
|---|---|---|---|---|---|
| # of search results | 27,846 | 29,860 | 31,926 | 500,717 | 590,349 |
| # of harbors (domain) | 4,807 | 2,515 | 3,878 | 27,713 | 38,913 |
| # of active harbors | 4,685 | 2,185 | 3,419 | 25,642 | 35,931 |
| # of postings | 532,413 | 640,073 | 1,469,251 | 6,497,263 | 9,139,000 |

importance of websites. A high PageRank score indicates a better reputation of the website, which can lead to a high search rank. To evaluate the overall quality of spam harbors, we use PageRank scores of spam harbors as an indicator of the quality of them. We randomly choose 1,000 spam harbors in each category, and use Google toolbar [8] to automatically request PageRank scores of these spam harbors. Figure 2 shows the PageRank scores distribution of these harbors.

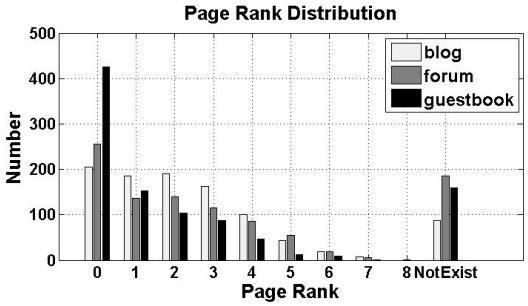



**Figure 2. PageRank Score Distribution of Harbors**

We can see that spammers target on both high-reputation and low-reputation harbors. From the graph, less than 20% harbors have a PageRank score higher than 3, which is the average PageRank score based on [18]. The reason is mainly because that websites with high PageRank scores usually have stronger spam sanitization mechanisms or more strict posting rules, which make it harder for spammers to keep automatically spamming on these websites. In addition, about 40% guestbook harbors have PageRank scores of 0, because most of them are small company websites that do not have notable reputations. However, spam links can still inherit and accumulate some reputation from a large number of such harbors. At least, they can use this way to let search engines to index them.

**Life Time** is defined as the time interval between the posting time of the first spam and the recent spam (based on our crawled dataset). Spammers tend to find some stable harbors that they can keep using. Thus, a long life time should be a good indication of high quality for spammers. Since there is no ground truth for the first spam and last spam, we randomly choose 100 harbors in each category and manually check their postings. Since we may not be able to crawl all the pages inside a given harbor, our estimated life time based on the limited crawled dataset is simply a lower bound. Figure 3 shows the distribution of life time of spam harbors.

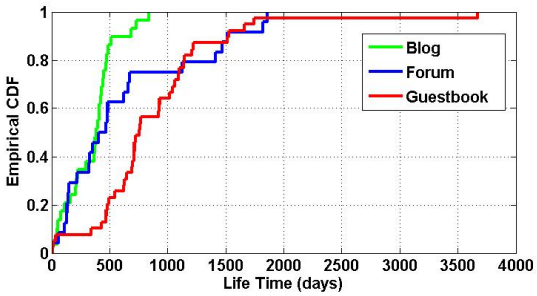



**Figure 3. Distribution of Harbor Life Time**

We can see that for both blog and forum harbors, more than 80% harbors have a long life time more than 1 year. And more than 70% guestbook harbors have a life time longer than 2 years. During the manual checking process, we found that for most harbors, the initial postings are benign but later these harbors are frequently exploited by spammers for spamming. Especially for guestbook harbors, almost all the later postings are spam, which confirms that these spam harbors are kept being used by spammers for a long time.

**Google Indexing Interval** is defined as the time difference between two consecutive Google crawling (indexing) time of the same spam harbor. To reduce the crawler overhead but also keep pace with dynamically updated web pages, search engine bots need to periodically crawl websites. Thus, there always exist a time lag between posting time and search engine indexing time. A shorter time lag (indexing interval) should be a sign of a high quality for spammers, because search engines can quickly index the new spam. Google Cache[7] contains the time when Google bot crawled the web page. Thus we randomly choose 100 active spam harbors in each category [3] and crawl their cache pages every day. Figure 4 shows the distribution of Google

[3]Note that Google Cache has a request limitation per day. Thus we only choose 100 harbors in this test. Also, Google does not cache all web pages, so we choose those pages that are cached by Google

indexing interval.

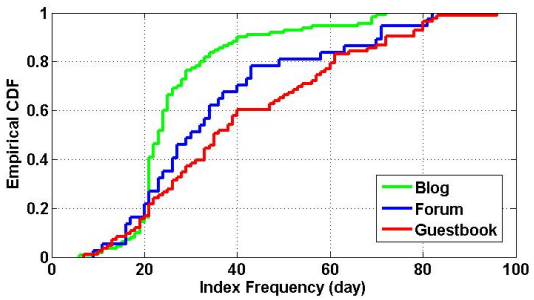



**Figure 4. Googe Indexing Interval**

Compared with guestbook harbors, blog and forum harbors have relatively shorter indexing intervals because normal postings on them update much more frequently than postings on guestbooks. Thus, Google bots crawl blog/forum harbors much more frequently than guestbook harbors. However, still nearly 80% of all harbors have an indexing interval larger than 20 days, which indicates that the overall indexing frequency is still not too high.

**Lessons learned:** Although high-reputation harbors should be the best choice for spammers, high-reputation websites usually have more strict spam filtering mechanisms or have stronger authentication systems, which makes it harder for spammers to automatically spam on them. Thus, spammers tend to *keep using* a large number of harbors for spamming *regardless of their reputations* to compensate the relatively poor quality of individual harbors. However, our observations also convey a positive message to the defenders: since there typically exists *a long time lag* between spamming time and search engine indexing time, if we can detect these spam before search engines index them, we can still efficiently prevent comment spamming from impacting search ranks.

## 3.3 Spam Harbors Infrastructure

After finding out qualified spam harbors, spammers intend to take full utilization of these harbors for spamming. In this section, we study how spammers utilize these harbors for spamming, and what are the relationships that spammers formed on these harbors.

### 3.3.1 Relationship Graph

To reduce the possibility of being detected, and also to take full utilization of their spam harbors, spammers tend to distribute their spam among multiple spam harbors. Thus, different spam harbors may *always* share similar spam, because spammers intend to recycle these harbors. This special close relationship among these harbors, which may rarely occur in the normal case, gives us a chance to study the spamming behaviors of spammers. To characterize such relationship, we build a relationship graph $G = (V, E)$ among these spam harbors. We view each spam harbor as a node $v$ and build up an edge $e$ between two spam harbors if they share same spam (links) in their postings. The resulting graph $G$ for our entire database consists of 13 connected sub-graphs, each of which has more than two nodes. The largest connected component $G_0$ contains 97 % spam harbor domains.

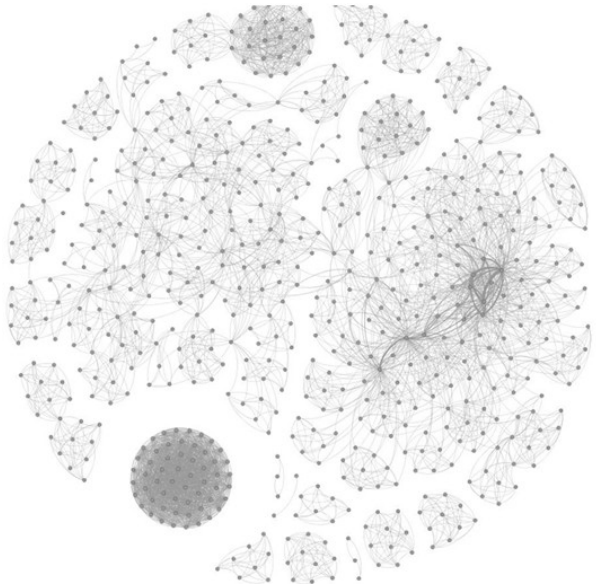


**Figure 5. Relation Graph of Spam Harbors**

Figure 5 is a partial visual representation of $G_0$. We can see that there exists a large number of *communities* within $G_0$, i.e., a group of nodes *closely* interconnected with each other and only loosely interconnected with other communities. Here, each community represents a set of harbors in close relationships with each other, possibly used by the same spammer. In addition, although different spammers may have different harbors, there always exist some harbors shared by multiple spammers, which provides us a good opportunity to find more other harbors (even starting from a small seed set).

### 3.3.2 Spam Harbor Community

In the spamming process, spammers first need to choose spam harbors to post their spam, and then need to distribute their spam on these selected harbors. In this section, we will study how spammers choose their harbors, and how they distribute spam to selected harbors.

**Choosing Spam Harbors.** In this part, we analyze how spammers choose spam harbors, i.e., we examine how many

harbors are used for spamming each time. Some spammers may spam on *all* their available harbors to fully use their resources, and some may only sample parts of their harbors for spamming to avoid the exposure/detection of all their resources. To measure how spammers choose harbors each time, we define a metric named "Distribution Ratio", which is the ratio of the number of harbors posting same spam over the number of harbors in their community [3]. Thus, a higher Distribution Ratio indicates that spammers tend to fully use their spam harbors for spamming. Figure 6 shows the distribution of Distribution Ratio for $S_{study}$.

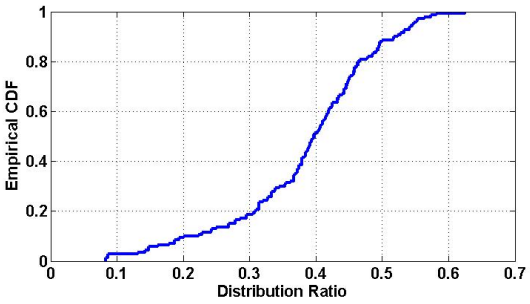



**Figure 6. Spam Distribution Ratio**

We can see that 80% of spammers tend to use only less than 50% of their spam harbors for the same spam. In this way, they can reduce the possibility of all their resources being detected/exposed. However, since spammers always have a limited number of harbors, to keep spamming, they have to recycle these harbors. As a result, we will finally observe a relatively stable relationship among harbors.

**Distributing Spam.** After selecting spam harbors, spammers need to decide how to distribute their spam to these selected harbors. For example, some spammers may post the same spam on their selected harbors at a similar time. In that case, posting time on these harbors should be similar. Other advanced spammers may choose to distribute different spam on different selected harbors. In this study, we simply consider two spam messages are posted in a similar time if they are posted in the same month. To measure the similarity of spam posting time, we design a metric, named "Time Centrality Ratio", which is the ratio of the maximal number of harbors that post the spam in the same month over the total number of harbors that post this spam. The intuition here is that if all the harbors post a spam message in the same month, it is possible that this spam is distributed to all selected harbors. Otherwise, spammers will distribute different spam to selected harbors. Thus, a high Time Centrality Ratio indicates that spammers distribute the same spam to most of their selected harbors at a similar time (some example is shown in Appendix A). Figure 7 shows the distribution of Time Centrality Ratio of $S_{study}$.

We can see that about 60% spam have a high ratio larger

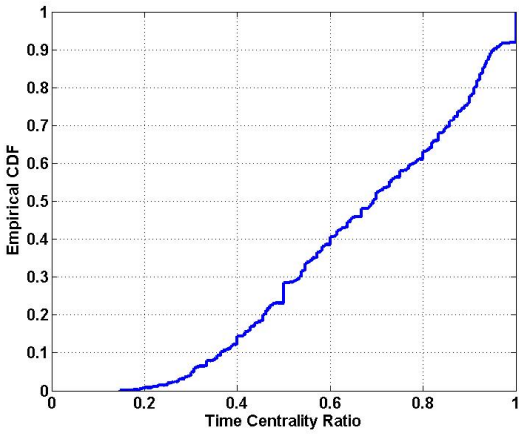



**Figure 7. Distribution of Time Centrality Ratio**

than 0.6. This means that about 60% spam is distributed by spammers to more than 60% of their selected harbors in one month for spamming.

**Lessons learned:** To efficiently utilize spam harbors, spammers intend to keep utilizing the spam harbors from a relatively stable set (pool) that they own. Thus, essentially spammers build an artificial relationship among these spam harbors, which is considered as their spamming structure. In addition, since spammers have a limited number of harbors, they must use/recycle these harbors with a large scale of spamming to maximize their profits. Also, although different spammers may have different strategies to find their harbors, there always exist some intersections among them, which gives us a chance to find more other spam communities even starting from a small seed set.

## 4 Inference System Design

In this section, we present a brief overview of our inference system, then describe in details its two core components: building spamming infrastructure graph and spam inference.

### 4.1 Overview

From the measurement study in Section 3, we can see that if a link (in a comment) is posted on a set of harbors that have a close relationship (e.g., within the same spam community in the infrastructure graph) at a similar time, it has a high possibility to be spam. Following this intuition, we design a spam inference system, named NEIGHBORWATCHER. NEIGHBORWATCHER infers comment spam purely based on the links promoted in the postings, ignoring other content information. An overview of NEIGH-

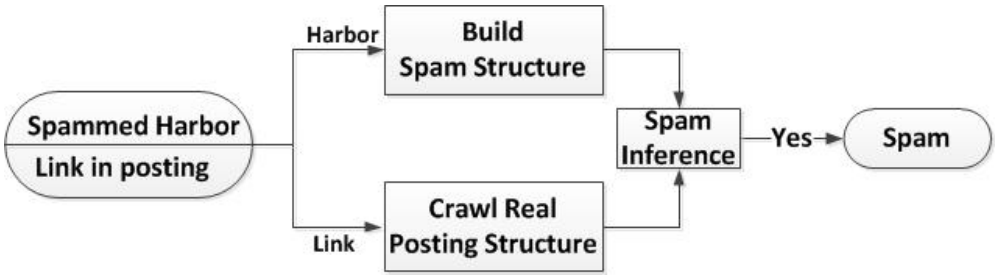



**Figure 8. System Architecture of NEIGHBORWATCHER**

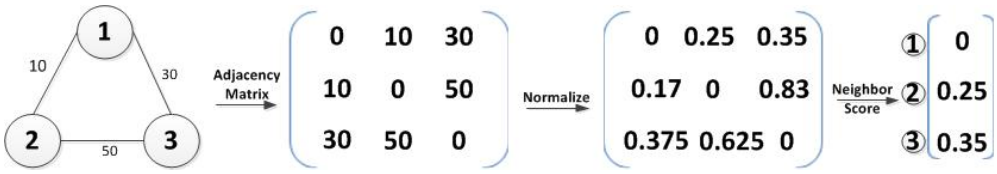



**Figure 9. Normalized Neighborhood Relationship Matrix**

BORWATCHER is shown in Figure 8. In practice, NEIGHBORWATCHER keeps monitoring (and updating) spam harbors in our database and builds the spamming infrastructure graph based on their posting history. In the inference phase, when a new post is given, NEIGHBORWATCHER extracts the embedded links, and also finds out the websites that have been posted with the same links (we call the set of these websites as a "real posting structure"). Based on the spam infrastructure and the real posting structure, NEIGHBORWATCHER calculates a suspicious score to tell how likely this is spam. Next, we will describe the algorithms of building our spamming infrastructure graph and inferring comment spam.

## 4.2 Building Spamming Infrastructure Graph

From Section 3, we know that spammers always have their own preferred spam harbors, and they intend to keep utilizing these spam harbors for spamming. Thus if multiple harbors always share similar postings in their history, it should be a good indication that they are exploited by the same spammer for spamming, and also have a high probability to be spammed by the same spammer in future. In this case, if we find a new posting occurs on these harbors at a similar time, we could infer this posting as spam with a reasonably high confidence. Following this intuition, we build spammers' spamming infrastructure based on shared postings (in historic data) among these spam harbors. We define the spamming infrastructure (or sometime we simply use "spamming structure" to denote the same concept) as neighborhood relationships among spam harbors. Thus, spam harbors spammed by the same spammers should have close neighborhood relationships, because they always share similar spam postings in history. To quantify such relationships, given an input harbor, we calculate neighborhood scores for all other spam harbors. A higher neighborhood score of a harbor indicates a much closer neighborhood relationship with the given (input) harbor.

To formalize the above intuition, we view all neighbor relationships among spam harbors as a weighted undirected graph $G = (V, E)$, in which $V$ denotes the set of all spam harbors, and each link $(i, j) \in E$ denotes that harbor $v_i$ and harbor $v_j$ share at least one common posting. The weight on the edge should reflect the strength of the relationship between two harbors. In our case, let $L_i$ be the set of postings (represented with their embedded URLs) in node $v_i$ and $L_j$ be the set of postings in node $v_j$, then we define weight $w_{i,j}$ as $|L_i \bigcap L_j|$. Thus, the more postings two harbors share, the much closer they are. We further normalize $w_{i,j}$ by dividing $\sum_j^n w_{i,j}$ as shown in Figure 9.

Next we design a graph-based propagation algorithm to propagate neighborhood score from the input harbor(s) to its neighbors based the neighborhood relationship graph $G$. Table 3 shows the notations used in our algorithms.

Before propagation, we first assign an initial score $I_i$ to each node $V_i$. For the input harbor $j$, $I_j$ is assigned with 1, and others are assigned with 0. Then we calculate neighborhood scores for all harbors as follows:

$$N = I \cdot W \tag{1}$$

Eq.(1) can capture the immediate neighbors of the input harbor. In this case, each immediate neighbor is assigned with a neighborhood score based on the number of common postings shared with the input harbor. The more common postings they share, the higher score they should have. As shown in Figure 9, node 3 has a higher score than node 1,

**Table 3. Notations Used in Our Paper**

| | |
|---|---|
| W | Normalized adjacency matrix of the neighbor graph |
| I | Input harbor vector, $I_i = 1$ if $i$ is a input harbor |
| N | Neighbor score vector. $N_i$ is the neighbor score of harbor $i$ |
| R | Real spam posting vector. $R_i = 1$ if harbor $i$ posts the same input link |
| $\alpha$ | Dampen factor. $\alpha = 0.85$ |
| n | The number of spam harbors |

because node 3 shares more common postings with the input harbor node 1 in history. However, as we show in Section 3, spammers might not always spam on all their harbors for each message, and our observed history relationships may be only a subset of the spammers' real relationships. To illustrate this scenario and demonstrate how we handle this problem in a generalized way, we show a case study in Figure 10.

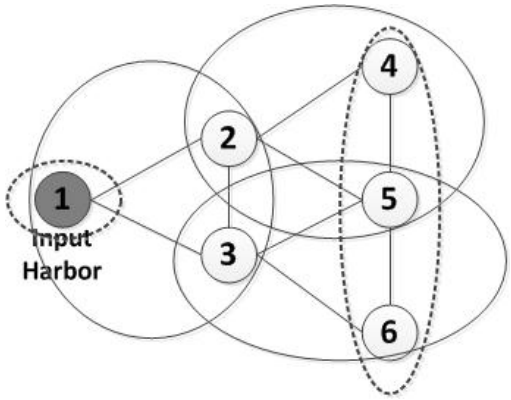



**Figure 10. A Case Study of Comment Spamming on Different Subsets of Harbors**

For this example, in the spamming process, a spammer first spams on node 1,2,3 for one spam message. And then the spammer spams on node 2,4,5 and node 3,5,6 with different spam messages. The neighborhood graph based on the history information is shown in solid circles. Now if the spammer spams on node 1, 4, 5, 6 (as seen in dashed circles), applying Eq.(1) with node 1 as the input harbor will assign score 0 to node 4, 5, 6, because they are not directly connected with the input harbor node 1. This makes neighborhood score less efficient to capture potential relationships between the input harbor and other harbors that will possibly be spammed by the same spammer. To overcome this problem, we need to deeply propagate the neighborhood relationship, similar to the page rank algorithm. Specifically, if we propagate the neighbor scores one further hop, this gives us $I \cdot W \cdot W = I \cdot W^2$, which will propagate neighbor scores from node 1 to 4. Thus the average received score for each node in this case is $(I\dot{W} + I \cdot W \cdot W)/2$. Naturally, the propagation scores should decay along with the distance from the input harbor. To achieve this goal, we dampen matrix $W$ by a constant $\alpha$ ( $0 < \alpha < 1$).[4] Thus the farther the distance between a harbor and the input harbor, the less neighbor score it can inherit. Based on all of above, we propagate neighborhood scores for each harbor as follows:

$$N = \frac{\sum_{i=1}^{t} I \cdot (\alpha \cdot W)^i}{t} \tag{2}$$

Once the neighbor score vector converges after $t$ propagation steps, we can obtain the final (stable) neighborhood scores for each harbor, and this score reflects how close the neighbor relationship is between the input harbor and the corresponding harbors. Next, we will present how to use these scores to infer a given new spam message.

### 4.3 Spam Inference

To infer whether a given new message/posting is spam or not, we also need to crawl other harbors to check if the link in the given posting also appears on them. Thus, we obtain a real posting structure vector $R$, $R_i = 1$ if harbor $i$ is also posted with the given link. Now we have both real posting structure and neighborhood scores for each harbor, next we present how to combine them to infer spam.

Intuitively, if harbors with high neighborhood scores have also been posted with the same messages, it has a high possibility that the input message is spam. Neighborhood scores reflect the neighbor relationships between other harbors and the input harbor. Thus, if harbors have high neighbor scores, they may have a high possibility to be spammed by the same spammer, which means if we find the input message appears in these harbors, it should have a high possibility to be spam. Thus, we infer the spam by computing how real posting structure and neighborhood scores combine together to contribute to a suspicious spam score. Specifically, we use a modified cosine similarity function $F(R, N)$ to characterize the similarity between the real posting structure and the learned neighborhood relationship in the spamming infrastructure. We

[4] We empirically set $\alpha = 0.85$ based on [21].

define the final spam score for the input message/URL $i$ as follows:

$$Score_i = F(R, N) = \frac{R \cdot N}{\sum_i^n R_i} \quad (3)$$

Here, a higher spam score means that the real posting structure matches very well with closer neighbors of the input harbor. Thus we should have a higher confidence to infer the input message as spam.

## 5 Evaluation

In this section, we evaluate our system in two stages. For the first stage, we evaluate NEIGHBORWATCHER regarding its inference stability and effectiveness. We also measure how many new spam and harbors can be inferred everyday, and the topic diversity of these spam postings. In the second stage, we discuss possible applications of our inference results.

### 5.1 Dataset and Ground Truth

To build the neighborhood relationship graph, we use the collected spam harbors in Section 3. After that, we keep monitoring these spam harbors everyday and extracting new postings from them for spam inference. Also, after inference, we search inferred spam links in Google and use the same way in Section 3 to extract new harbors from search results. To evaluate the effectiveness of our inference system, we need to choose true spam links and benign links for testing. For the former, we extract random postings from harbors and manually choose 500 verified spam messages (that contain spam links). To find a normal postings set, we assume domains in Alexa [1] top 20,000 are highly-reputable websites that have less chance to be posted in comment spam. Thus, we check how many links in our collected postings have intersection with these domains. In this way, we get 754 normal postings and combine them with 500 spam postings as our testing dataset $S_{test}$.

### 5.2 Stability of Spamming Structure

Our system exploits spammers' posting infrastructure (or spamming structure) to infer spam. Thus, if such infrastructure changes frequently, it will make our system less effective. For example, if spammers keep using new harbors for spamming everyday, our system cannot infer their spam. To evaluate the stability of spamming structure, we essentially examine how the neighborhood relationship among spam harbors change over the time. We build the neighborhood relationship graph of spam harbors by setting $w_{i,j} = 1$ to indicate that two spam harbors share at least one common link and $w_{i,j} = 0$ to represent no relationship between two harbors. Then we consider such relationship graph in the time window $[t, t + \Delta t^5]$ as the initial spamming structure and the relationship graph in the next window $[t + \Delta t, t + 2\Delta t]$ as the testing spamming structure. Thus, the difference between two structures indicates the instability of the spamming structure. To quantify such instability, we use Hammin Distance [12] to measure the number of changed relationships $CR$ between these two time window, as shown in Eq.(4). Here $n$ is the number of total harbors.

$$CR_i = \sum_j^n |W_{i,j}^{t+2\Delta t} - W_{i,j}^{t+\Delta t}| \quad (4)$$

Thus, a smaller value of $CR$ implies a much more stable spamming structure. For each spam harbor $i$ in our database, we calculate its changed relationship $CR_i$. Figure 11 shows the distribution of changed relationships for all spam harbors.

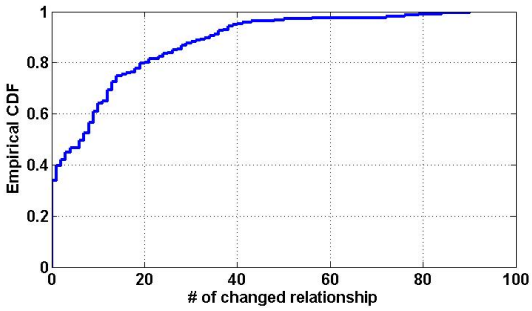



**Figure 11. Changed Relationship Distribution**

We can see that about 40% harbors do not change their neighbor relationships because spammers keep utilizing the same harbors. In addition, about 80% spam harbors change their relationships less than 20, which is also less than half of the average community/clique size 50. Thus, even if a community loses 20 harbors, we can still infer spam with the remaining 30 harbors as long as spammers keep recycling their harbors. Furthermore, the continuous updating of our harbor database can somehow compensate such instability, we will discuss more about this in Section 6.

### 5.3 Effectiveness of Inference

To evaluate the effectiveness of our system, we test our system with $S_{test}$. We consider both the number of correctly inferred spam, termed as "Hit Count", and the ratio of this number to the total number of inferred spam, termed as "Hit Rate" (i.e., accuracy). Thus, a higher value of Hit Count indicates that our system can catch more spam; and a higher value of Hit Rate indicates that our system can infer spam more accurately.

[5] We empirically set $\Delta t$ 1 month here.

**Table 4. Sensitivity of the Hit Rate and Hit Count to the Choice of Similarity Threshold**

| Sim. Threshold | Hit Rate | Hit Count | False Positive |
|---|---|---|---|
| 0.3 | 57.29% | 432 | 322 |
| 0.4 | 75.93% | 426 | 135 |
| **0.5** | **97.14%** | **408** | **12** |
| 0.6 | 97.8% | 360 | 8 |

Since we infer spam based on the spam score threshold, we also check how the threshold contributes to the inference results, a way similar to [19]. In our case, a higher (thus more conservative) threshold may lead to a higher hit rate but with a lower hit count. Table 4 shows how Hit Rate and Hit Count vary with different settings of the threshold. We can see that a spam score threshold of 0.5 yields to a relatively high hit rate (97%) with a relatively high hit count. Also, there are still 92 links that can not be correctly inferred as spam (i.e., false negatives) and 12 incorrect inferred links based on our labels (i.e., false positives). We further check these links, most of false negatives only appeared in its input harbor, which means these spam links do not appear in other harbors in our database. This is possible because our database is relatively small and may not cover all spammers' harbors. However, the effectiveness could be easily further improved if one can build a larger dataset, e.g., by recursively updating the spam harbors database, which will be discussed in Section 6. Among 12 false positives, 5 are actual benign websites posted by spammers in order to conduct harbor testing (as discussed in Figure 1). 7 of them are those link to some Alexa top 20,000 websites and we expected (labeled) them to be non-spam as explained in our test dataset preparation. However, they turn out to be actual spam posted on reputable websites (e.g., in some Google groups). If we exclude them, our actual false positives are only five, which is pretty low. Finally, we note again that our inference algorithm only uses the spamming structure information, and it does not leverage other existing content features yet. Once combined with existing features, we surely can expect a better performance.

## 5.4 Constancy

To evaluate the constancy of our system, we essentially examine whether our system can continue finding new spam over time. We keep monitoring spam harbors everyday to check new postings. These new postings are submitted to our system for spam inference. For each new inferred spam, we further search it in Google to find new spam harbors (using the same method mentioned in Section 3) that are not included in our database, then add them in our database everyday.[6] After 3 weeks' study, we have totally inferred 91,636 new spam and 16,694 new spam harbors. Figure 12 is the distribution of new inferred spam and new spam harbors over time. We can see that our system can constantly find new spam and spam harbors as long as spammers keep posting new spam and also spamming on new harbors.

**Diversity of New Spam.** To have a sense of the variety of spam content we inferred, we surveyed 10,000 randomly chosen spam postings and clustered them in 7 categories based on their anchor keywords. Table 5 shows the keywords we used for clustering spam, and Figure13 shows the category results. We can see that pharmacy is still the most popular spam, and spammers always try to promote them to have a higher search rank [30]. On the other hand, our system can keep capturing general spam (other than just pharmacy) in terms of their spam content.

**Table 5. Keywords for Different Spam Category**

| Categary | Terms |
|---|---|
| Rogue Pharmacy | cialis,viagra,prescription,levitra ... |
| Rogue software | virus,windows,desktop,software... |
| Porn | porn,sexy,fuck,adult... |
| Gambling | casino,poker,roulette,gambling... |
| Money | insurance,debt,mortgage,credit... |
| Accessories | handbag,dress,luxurious,louis... |

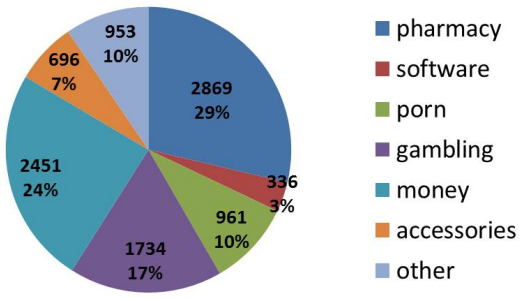
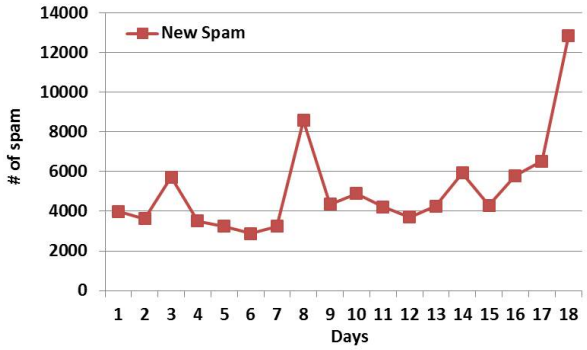


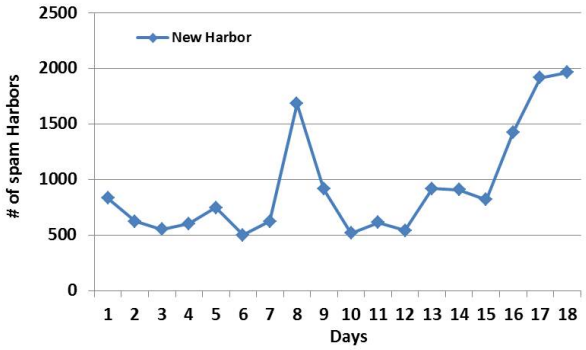

**Figure 13. Spam Category**

**Context-based Analysis of Spam.** For those newly inferred spam, we randomly sample 1,000 spam links and use the same method in [32] to find out URL-redirection and cloaking spam. Specifically, we send requests to each link 3 times by setting different HTTP header parameters

[6] We may add a few normal websites in our database in this way because our inference hit rate is not 100%. However, we note that these websites most likely will not have close relationships with other spam harbors, thus will not impact our inference results much.




(a) New Spam




(b) New Spam Harbors

**Figure 12. Constancy of NEIGHBORWATCHER**

to emulate Google Bot crawling, directly visiting, and visiting through search result clicking, respectively. We consider it as cloaking spam if any of two visits lead to different websites through redirection. In this case, among 1,000 spam links, we only see 34 spam using clocking techniques, which also reflects that context-based detection has a low coverage in face of current comment spam. However, we test these 34 spam links with Google Safe Browsing (GSB) [10], none of them has been reported. Thus context-based detection is still an effective way to find new spam within their limited scope, and our system also can cover such spam (but can detect more other spam that the context-based detection can not).

### 5.5 Applications of Our System

In this part, we will evaluate how our inference results can be applied to provide early warning for search engine bots, and to complement current BlackList services.

**Early Warning.** "nofollow"[13] is a HTML tag which is designed to instruct search engines that the corresponding links should not be counted for ranking. Usually different search engines have a little different policies in face of "notfollow" tags. As for Google [14], it will not transfer PageRank or anchor text across corresponding links. In this case, search engines can efficiently filter comment spam if webmasters attach each posting with a "nofollow" tag. However, among 35,931 spam harbors we found, only 4,367 harbors contain such tags, which means spam on other harbors can be successfully exploited for search rank manipulation. Fortunately, according to our measurement study in Section 3, there always exists a time lag between the time that spammers post spam and the time search engines index the spam. Thus, if we can detect the spam before Google indexes them, we can also efficiently filter comment spam. To measure this timeliness, we examine the number of "zero day spam", which is the spam that can not be searched out by Google at the time. Totally we collected 1,364 "zero day spam" in our test using NEIGHBORWATCHER. Interestingly, when we manually check these "zero day spam", we find that some spam messages contain randomly generated domains that have not been registered yet. Thus, it is possible that spammers may first promote these links and then register them later based on their promoting results. Figure 14 shows the distribution of daily "zero day spam".

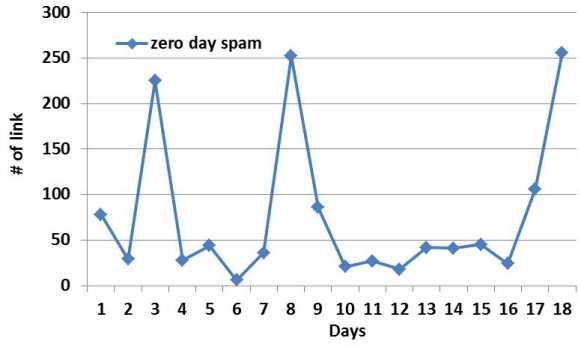



**Figure 14. Zero Day Spam Distribution**

We can see that currently we can only detect few "zero day spam", because most spammers intend to promote certain set of spam links that may have already been indexed by Google before (but search ranks were probably not good enough). However, as long as spammers begin to promote new links, our system can quickly find more "zero day spam". In this case, we can give an early warning to search engine bots, or search engine bots can integrate our inference system to help better filter these comment spam.

**BlackList.** Existing comment spam blacklists usually provide a collection of source IP addresses, register emails, or usernames of spammers. Most of existing online blacklists [17, 16, 5] collect such spammer information by building honey blogs/forums. However, we can early imagine

that honey blogs could only collect a limited number of spam, thus limiting the effectiveness of such approaches.

To measure our inference approach can complement existing solutions, we compare it with 3 popular online BlackList services. Note that since our system targets on more general spam harbors, not all of the harbors provide complete IP and email information of the posting users in public. Luckily, there does exist some spam harbors in our database that provide IP or email address information. Thus, we can compare these inferred IPs and emails with these 3 Blacklist services.

Table 6 shows the comparison result. We can see that for both IP and email, our system can always infer new spammers that are not observed by existing services. Figure 15 shows the daily comparison with these 3 BlackList Services. From the figure, each day we can find new spammers that are not labeled by any of these existing BlackLists. In addition, considering the dynamic property of IP addresses, most IP-based BlackLists need to be daily updated. Thus, for IPs detected by these existing systems, we further check their "last seen time", the time of last observation by these existing services. We find most of them are out of date, which means existing BlackLists observe these spammers long time ago but in fact they are still active at the moment. In summary, our system could be a good complement to existing BlackList systems to actively find new spam and to improve the coverage of existing systems. Furthermore, we find some constant email addresses and IPs keep contributing to spam, which also reflects that spammers intend to keep utilizing these spam harbors.

**Table 6. Comparison with Existing Blacklist Systems**

| BlackList | # of IP | # of Email |
|---|---|---|
| NeighbourWatcher | 378 | 5,945 |
| StopForum | 231 | 1,937 |
| SpamBust | 185 | 122 |
| GlobalSpyware | 29 | 3 |

# 6 Discussion

Although NEIGHBORWATCHER shows promise for inferring new spam and spam harbors, there is still much room for improvement. In this section, we discuss several specific improvement areas, and the possible evasions to our current inference algorithm.

## 6.1 Improvement

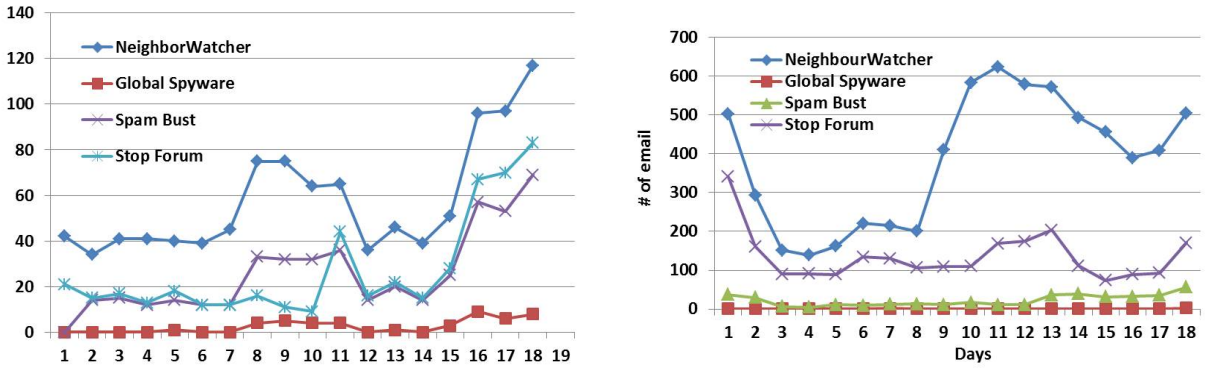
**Combining More Features.** NEIGHBORWATCHER only uses spamming structure information to infer spam, other spamming behaviors could also be incorporated together to help improve the accuracy. For example, we find some spammers post their spam with both $< a\ href = $ "..." $>$ tags and BBcode tags to assure they can embed spam links, because they have no idea about what kind of method the target spam harbors support to embed links. In addition, some spam links are posted on websites with different languages because spammers do not care about harbor's native languages. For example, some spammers post Russian spam in Chinese websites, Korean websites, and English websites, which is extremely unlikely to be normal postings. In this case, we can incorporate these features to improve the overall inference accuracy.

**Improving Algorithms.** In Eq.(3), we assume these postings on different harbors have the same weight. However, postings on different harbors are not necessary equal. For example, if postings on a harbor have a similar posting time, same email, or same IP address with the input posting, then it has a high possibility that they are spammed by a same spammer at the same time. Thus we could assign different weight to different harbors by considering these factors.

**Updating Spam Harbors.** In our system, we find new spam harbors by recursively searching inferred spam in search engines. Thus, as long as spammers keep utilizing these spam infrastructures, we can always find out new spam harbors. However, our current system cannot infer those spam posted on only one harbor in our dataset, thus we cannot find out other harbors that are also posted with this spam. In this case, we could use the following strategy. From the search results of the posting link, we attempt to build a relationship graph among returned websites based on whether they already share similar postings (excluding the searched link) on existing pages on the websites. If these websites show very close relationships (e.g., dense connections), the searched link has a good chance to be spam (and thus the corresponding search result websites are possible spam harbors). The intuition here is that although a few normal links may appear on a variety number of websites, it is extremely unlikely that normal users will *keep* posting *similar* postings on certain websites. Our ongoing work will design and test new algorithms to efficiently update our spam harbors dataset.

## 6.2 Possible Evasion

**Evasion by exploring new harbors.** Obviously spam harbors that we collected are only the subset of spammers' harbors. Thus if spammers know our collected harbors, they




(a) Daily IP (b) Daily Email

**Figure 15. Daily Comparison with Existing Spam Blacklists**

may try to spam on other harbors that are not included in our database, or they may find brand new harbors. In this case, our neighborhood-based inference algorithm could not detect their new structure. However, as long as spammers also post spam on both these new harbors and old harbors, we can still find out their new harbors by keeping updating our database. Otherwise, spammers need to keep finding new harbors and give up existing qualified harbors, which is less likely to happen considering the cost.

**Evasion by changing spamming behaviors.** If spammers know that we use spam links to build up relationship graphs, spammers may spam different links on different harbors. Thus we can not build up their spamming structure. However, in this case, spammers need to keep finding more harbors to post their variety links, which will increase their cost and time. Otherwise, they need to post the same spam several times on certain harbors (in order to boost search ranks), which will increase the possibility of being detected by content-based detection systems.

**Evasion by spamming polymorphic URLs.** Our algorithm relies on grouping identical URLs (e.g. we infer a possible spam message if the spamming URL also appears on many of its neighborhood clique harbors). Thus, spammers may try to evade our system with polymorphic URLs (i.e., each URL can be different on different harbors). However, in general, it is not always possible to make full polymorphic URLs for a given spam URL to be promoted. If spammers choose URL shortening services to achieve polymorphic URLs, we can always use the resolved final URLs in our system. Furthermore, we can use the domain of a spam URL instead of the full URL as input, which is relatively stable if spammers want to promote certain domains.

## 7 Related Work

In this section, we discuss related work in the following two perspectives.

**Comment Spam:** To detect comment spam, a few studies have been done previously. Kolari *et al.* [31] proposed to detect blog spam based on only content. They use a machine-learning based method to detect spam by extracting features from the posting content, such as bag-of-words, bag-of-anchors, and bag-of-URLs. Mishne *et al.* [24] proposed a language model based approach to detect comment spam. They generate a language model for the blog posts, the comments, and the pages linked by spam, and then use language model divergences to classify spam.

Recently, to overcome the pitfalls of content-based detection systems, Niu *et al.* [32] studied comment spam in both legitimate and honeypot forums, and proposed a context-based detection method that looks for redirection and cloaking for forum spam. Shin *et al.* [33] studied the characteristics of spam from a research blog for 286 days, and developed a light-weight SVM classifier to detect forum spam. Their features include 16 different features extracted from spammer origins, comment activities, URLs in spam comments, and textual contents. Tan *et al.* [23] conducted a comprehensive analysis of spamming activities on a commercial blog site covering over 6 million posts and 400,000 users. Through studying on non-textural patterns, such as posting activities, spam link metrics, they developed a realtime system to detect spam postings based on 13 non-textual features. Kantchelian *et al.* [28] define spam as uninformative content, and thus they proposed a machine-learning based method to detect spam based on the content complexity of comments in social medias.

Most of these existing research studies the problem from certain blogs or honeypot blogs. However, we study from a large number of spam harbors that are frequently used

for spamming. We present an in-depth study of these harbors, which could not be observed from only a few blogs. Furthermore, complementary to content-, context-, and behavior-based detection features, our system exploits the spamming infrastructure to detect comment spam, which is much more stable in the spamming process. Thus our system is a good supplement to existing research.

**Graph-based algorithm:** Graph-based algorithms have been previously applied in spam detection. Pagerank [29] and Trustrank [35] have been widely used by search engines to determine the search ranks of search results. Zhao *et al.* [22] proposed a spam detection system by exploring link dependencies and content similarities among web pages. The system first clusters the hosts based on content features and assigns labels for clusters using majority voting. Then the system propagates these labels to their neighbor hosts and uses predicted labels as new features to retrain the classifier. Different from these work, our paper uses a different graph-based algorithm to characterize the spamming structure, and combines with the real posting structure to infer spam.

Our inference algorithms are motivated by the following two studies [26, 20]. Ramachandran *et al.* [20] proposed an email spam filtering system. The system takes email-sending patterns of all senders as input and builds clusters of sending patterns from a small seed of spammers. Thus, it classifies senders as spammers if their behaviors are similar to the patterns of known spammers. Zhang *et al.* [26] built a system to predict blacklist based on attackers' history. Their intuition is that if two hosts are frequently attacked by an attacker in history, it has a high probability that they will be attacked by the same attacker in the future. Our work shares a similar intuition with these two studies, however in a totally different application and with different inference algorithms. Furthermore, we present a measurement study of spam harbors, which is not shown by any prior work.

## 8 Conclusion

Although comment spam has been studied for several years, arms race between spammers and researchers has made current existing detection systems losing their potency. A lots of spamming techniques are developed by spammers to evade content- or context-based detection systems.

In this paper, we present a deep study on comment spam from a new perspective, i.e., the spamming infrastructure, which is the core and stable part in the comment spamming process. Through measuring 35,931 spam harbors exploited by spammers to post spam, we conclude that spammers prefer to *keep utilizing* their spam harbors for spamming unless they are blocked. Based on this finding, we design a graph-structure-based inference system to infer comment spam by checking if the same spam also appears on its neighbor (clique) harbors. Our evaluation results show that we can infer a large number of new comment spam and spam harbors, and keep finding them everyday.

## 9 Acknowledgments

This material is based upon work supported in part by the National Science Foundation under Grant CNS-1218929. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

## References

[1] Alexa rank. http://www.alexa.com/.

[2] Botnets and google dorks: A new recipe for hacking. http://www.darkreading.com/vulnerabilitymanagement/167901026/security/vulnerabilities/231500104/botnetsandgoogledorksanewrecipeforhacking.html.

[3] cfinder. http://www.cfinder.org/.

[4] The (evil) genius of comment spammers. http://www.wired.com/wired/archive/12.03/google.html?pg=7.

[5] Globalspyware. http://globalspyware.com/.

[6] Google bombs. http://www.searchenginepeople.com/blog/incredible-google-bombs.html.

[7] Google cache. http://www.googleguide.com/cached_pages.html.

[8] Google pagerank api in php. http://www.fusionswift.com/2011/10/google-pagerank-api-in-php-october-2011/.

[9] Google rolls out content spam detection. http://www.nationalpositions.com/blog/seonewsgooglerollsoutcontentspamdetection/.

[10] Google safe browsing. http://code.google.com/apis/safebrowsing.

[11] Google search and search engine spam. http://googleblog.blogspot.com/2011/01/google-search-and-search-engine-spam.html.

[12] Hamming distance. http://en.wikipedia.org/wiki/Hamming_distance.

[13] Notfollow. http://en.wikipedia.org/wiki/Nofollow.

[14] rel="nofollow". http://support.google.com/webmasters/bin/answer.py?hl=en&answer=96569.

[15] Safe browsing-protecting web users for five years and counting. http://googlepublicpolicy.blogspot.com/2012/06/safe-browsingprotecting-web-users-for.html.

[16] Spambust. http://spambusted.com/.

[17] Stop forum spam. http://www.stopforumspam.com/.

[18] What does your google pagerank mean. http://www.redfusionmedia.com/google_pagerank.htm.

[19] K. W. A. Ramachandran, A. Dasgupta and N. Feamster. Spam or ham?: characterizing and detecting fraudulent not spam reports in web mail systems. In *Proceedings of the 8th Annual Collaboration, Electronic messaging, Anti-Abuse and Spam Conference(CEAS 11)*, 2011.

[20] N. F. A. Ramachandran and S. Vempala. Filtering spam with behavioral blacklisting. In *Proceedings of the 14th ACM conference on computer and communications security,*, 2007.

[21] S. Brin and L. Page. The anatomy of a large-scale hypertextual Web search engine. In *Proceedings of the seventh international conference on World Wide Web*, 1998.

[22] D. D. C. Castillo and A. Gionis. Know your neighbors: Web spam detection using the web topology. In *ACM Special Interest Group on Information Retrieval (SIGIR) Conference, July*, 2007.

[23] S. C. X. Z. E. Tan, L. Guo and Y. Zhao. Spam behavior analysis and detection in user generated content on social network. In *Proceedings of 32nd International Conference on Distributed Computing Systems (ICDCS 2012), Macao, China, June 18-21,*, 2012.

[24] D. C. G. Mishne and R. Lempel. Blocking Blog Spam with Language Model Disagreement. In *First International Workshop on Adversarial Information Retrieval on the Web, at 14th international conference on World Wide Web(WWW)*, 2005.

[25] Z. Gyongyi and H. Garcia-Molina. Web Spam Taxonomy. In *Technical report, Stanford Digital Library Technologies Project, Mar*, 2004.

[26] P. P. J. Zhang and J. Ullrich. Highly Predictive Blacklisting. In *Proceedings of the USENIX Security Symposium, San Jose, CA, July*, 2008.

[27] Z. X. J. Zhang, C. Y and G. Gu. PoisonAmplifier: A Guided Approach of Discovering Compromised Websites through Reversing Search Poisoning Attacks. In *Proceedings of the 15th International Symposium on Research in Attacks, Intrusions and Defenses (RAID'12)*, 2012.

[28] A. Kantchelian, J. Ma, L. Huang, S. Afroz, A. Joseph, and J. D. Tygar. Robust detection of comment spam using entropy rate. In *Proceedings of the 5th ACM workshop on Security and artificial intelligence (AISec'12)*, 2012.

[29] R. M. L. Page, S. Brin and T. Winograd. The PageRank citation ranking: Bringing order to the Web. In *Technical report, Stanford University Database Group, http://citeseer.nj.nec.com/368196.html*, 1998.

[30] T. M. N. Leontiadis and N. Christin. Measuring and analyzing search-redirection attacks in the illicit online prescription drug trade. In *Proceedings of the 20th USENIX Security*, 2011.

[31] T. F. P. Kolari and A. Joshi. SVMs for the blogosphere: Blog identification and splog detection. In *Proceedings of AAAI Spring Symposium on Computational Approaches to Analysing Weblogs, March*, 2006.

[32] H. C. M. M. Y. Niu, Y.M. Wang and F. Hsu. A quantitative study of forum spamming using context-based analysis. In *Proceedings of Network and Distributed System Security Symposium(NDSS), February*, 2007.

[33] M. G. Y. Shin and S. Myers. Prevalence and mitigation of forum spamming. In *Proceedings of the 30th Annual IEEE Conference on Computer Communications (INFOCOM*, 2011.

[34] M. G. Y. Shin and S. Myers. The Nuts and Bolts of a Forum Spam Automator. In *Proceedings of the Wkshp. on Large-Scale Exploits and Emergent Threats (LEET)*, 2011.

[35] H. G.-M. Z. Gyongyi and J. Pedersen. Combating Web Spam with TrustRank. In *30th International Conference on Very Large Data Bases, Aug.*, 2004.

# A Spam Search Result Example

Figure 16 shows an example of a Google search result page for a spam link. We can draw the following conclusions from it: (1) Spammers do use a variety of spam harbors for spamming. (2) Spammers post this spam in different harbors at a similar time (Mar 23, 2012). (3) Spammers post this spam with the similar anchor text:"allgrannysex".

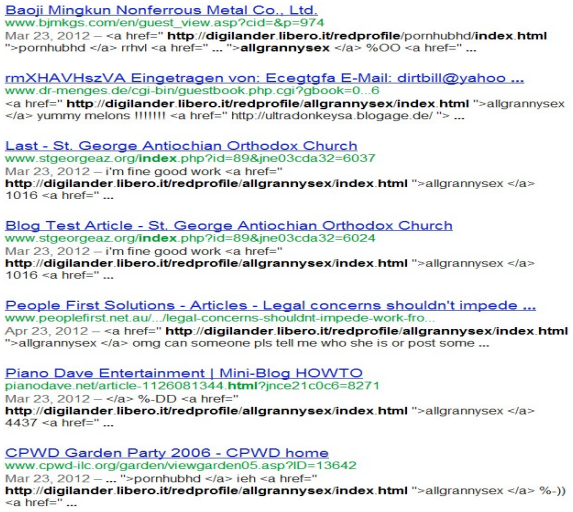



**Figure 16. Example of Spam Search Results**