deSEO: Combating Search-Result Poisoning

John P. John, [‡*] *Fang Yu*[§]*, Yinglian Xie*[§]*, Arvind Krishnamurthy*[‡]*, Martín Abadi*[§†]
[‡]*University of Washington* [§] *MSR Silicon Valley*
{*jjohn, arvind*}*@cs.washington.edu* {*fangyu, yxie, abadi*}*@microsoft.com*

Abstract

We perform an in-depth study of SEO attacks that spread malware by poisoning search results for popular queries. Such attacks, although recent, appear to be both widespread and effective. They compromise legitimate Web sites and generate a large number of fake pages targeting trendy keywords. We first dissect one example attack that affects over 5,000 Web domains and attracts over 81,000 user visits. Further, we develop deSEO, a system that automatically detects these attacks. Using large datasets with hundreds of billions of URLs, deSEO successfully identifies multiple malicious SEO campaigns. In particular, applying the URL signatures derived from deSEO, we find 36% of sampled searches to Google and Bing contain at least one malicious link in the top results at the time of our experiment.

1 Introduction

The spread of malware through the Internet has increased dramatically over the past few years. Along with traditional techniques for spreading malware (such as through links or attachments in spam emails), attackers are constantly devising newer and more sophisticated methods to infect users. A technique that has been gaining prevalence of late is the use of search engines as a medium for distributing malware. By gaming the ranking algorithms used by search engines through search engine optimization (SEO) techniques, attackers are able to poison the search results for popular terms so that these results include links to malicious pages.

A recent study reported that 22.4% of Google searches contain such links in the top 100 results [23]. Furthermore, it has been estimated that over 50% of popular keyword searches (such as queries in Google Trends [9] or for trending topics on Twitter [20]), the very first page of results contains at least one link to a malicious page [19].

Using search engines is attractive to attackers because of its low cost and its legitimate appearance. Malicious pages are typically hosted on compromised Web servers, which are effectively free resources for the attackers. As long as these malicious pages look relevant to search engines, they will be indexed and presented to end users. Additionally, users usually trust search engines and often click on search results without hesitation, whereas they would be wary of clicking on links that appear in unsolicited spam emails. It is therefore not surprising that, despite being a relatively new form of attack, search-result poisoning is already a huge phenomenon and has affected major search engines.

In this paper, we aim to uncover the mechanics of such attacks and answer questions such as how attackers compromise a large number of Web sites, how they automatically generate content that looks relevant to search engines, and how they promote their malicious pages to appear at the top of the search results.

In order to answer these questions, we examine a live, large-scale search poisoning attack and study the methods used by the attackers. This attack employs over 5,000 compromised Web sites and poisons more than 20,000 popular search terms over the course of several months. We investigate the files and scripts that attackers put up on these compromised servers and reverse-engineer how the malicious pages were generated.

Our study suggests that there are two important requirements for a search-result poisoning attack to be successful: the use of multiple (trendy) keywords and the automatic generation of relevant content across a large number of pages. Since trendy keywords are often popular search terms, poisoning their search results can affect a large user population. Further, by generating many fake pages targeting different keywords, attackers can effectively increase their attack coverage.

Based on these observations, we develop techniques to automatically detect search-result poisoning attacks. Although there exist methods for identifying malicious content in individual Web pages [14, 22], these solutions are not scalable when applied to tens of billions of Web pages. Further, attackers can leverage *cloaking* techniques to display different content based on who is requesting the page—malicious content to real users and benign, search-engine-optimized content to search engine crawlers. Therefore, instead of detecting individual SEO pages, we identify groups of suspicious URLs—typically containing multiple trendy keywords in each URL and exhibiting patterns that deviate from other URLs in the same domain. This approach not only is more robust than examining individual URLs, but also can help identify malicious pages without crawling and evaluating their actual contents.

Using this approach, we build *deSEO*, a system that automatically detects search-result poisoning attacks

[*] Work partly performed while interning at MSR Silicon Valley.
[†] Also affiliated with UC Santa Cruz and Collège de France.

without crawling the contents of Web pages. We apply deSEO to two datasets containing hundreds of billions of URLs collected at different periods from Bing. Our key results are:

1. deSEO detects multiple groups of malicious URLs, with each malicious group corresponding to an SEO campaign affecting thousands of URLs.
2. deSEO is able to detect SEO campaigns that employ sophisticated techniques such as cloaking and have varying link structures.
3. We derive regular expression signatures from detected malicious URL groups and apply them to search results on Google and Bing. The signatures detect malicious links in the results to 36% of the searches. At the time our experiments, these links were not blocked by either the Google Safebrowsing API or Internet Explorer.

The rest of the paper is structured as follows. We begin with describing the background for SEO attacks and reviewing related work in Section 2. Next, we investigate a large scale attack in detail in Section 3. Based on the insights gained from the attack analysis, we present the deSEO detection system in Section 4. In Section 5, we apply deSEO to large datasets and report the results. We analyze the detected SEO groups and apply the derived signatures to filter search results in Section 6. Finally, we conclude in Section 7.

2 Background and Related Work

Search engines index billions of pages on the Web. Many modern search engines use variants of the PageRank algorithm [17] to rank the Web pages in its search index. The rank of a page depends on the number of incoming links, and also on the ranks of the pages where the links are seen. Intuitively, the page rank represents the likelihood that a user randomly clicking on links will end up at that page.

In addition to the rank of the page, search engines also use features on the page to determine its relevance to queries. In order to prevent spammers from gaming the system, search engines do not officially disclose the exact features used to determine the rank and relevance. However, researchers estimate that over 200 features are used [3, 6]. Among these features, the most widely known ones are the words in the title, the URL, and the content of the page. The words in the title and in the URL are given high weight because they usually summarize the content of the page.

Search Engine Optimization (SEO) is the process of optimizing Web pages so that they are ranked higher by search engines. SEO techniques can be classified as being *white-hat* or *black-hat*.

In white-hat SEO, the sites are created primarily with the end-user in mind, but structured so that search engine crawlers can easily navigate the site. Some of the white-hat techniques are creating a sitemap, having appropriate headings and subheadings, etc. They follow the quality guidelines recommended by search engines [8, 29].

Black-hat SEO techniques, on the other hand, try to game the rankings, and do not follow the search engine guidelines. Keyword stuffing (filling the page with lots of irrelevant keywords), hidden text and links, cloaking (providing different content to crawlers and users), redirects, and participating in link farms are considered black-hat techniques. These practices are frowned upon by the search engines, and if a site is caught using such techniques, it could be removed from the search index.

To detect black-hat SEO pages, many approaches have been proposed. Some are based on the content of the pages [15, 5, 21], some are based on the presence of cloaking [25, 27], while some others are based on the link structure leading to the pages [26, 4].

The SEO attacks that we study in this paper are different from traditional ones in that attackers leverage a large number of compromised servers. Since these servers were originally legitimate and their main sites still operate normally even after compromise, they display a mixed behavior and therefore are harder to detect.

Our detection methods make use of URL properties to detect malicious pages without necessarily crawling the pages. In this respect, our work is similar to previous work by Ma *et al.* [13, 12], where they build a binary classifier to identify email spam URLs without crawling the corresponding pages. The classifier uses training data from spam emails. The SEO attacks we study are very new and there are few reports on specific instances of such attacks [10]. Therefore, it is difficult to get training data that has good coverage. In addition, spam URLs have different properties than SEO URLs. Many spam domains are new and also change DNS servers frequently. Therefore, their system makes use of domain-level features such as age of the domain and the DNS-server location. Since we deal with compromised domains, there are no such strong features.

A recent analysis of over 200 million Web pages by Google's malware detection infrastructure discovered nearly 11,000 domains that are being used to serve malware in the form of FakeAV software [18]. This work looks at the prevalence and growth of FakeAV as a means for delivering malware. Our work, on the other hand, looks at the mechanisms used by the perpetrators to game search engines for the effective delivery of this kind of malware. By developing methods to detect SEO attacks, we also detect a large number of compromised domains, but without having to inspect them individually.

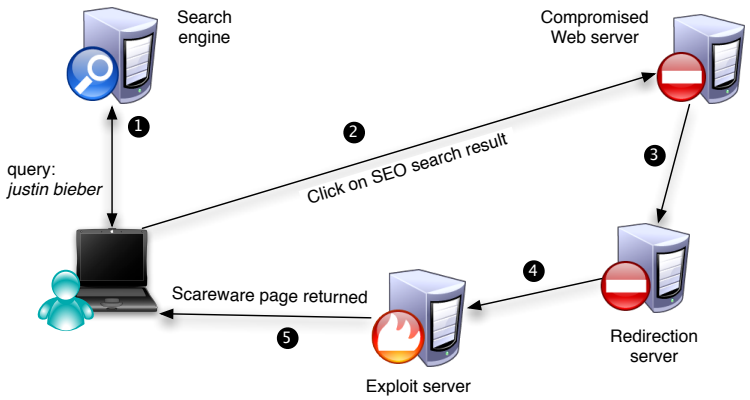

Figure 1: An overview of how the attack works. The victim issues a popular query to a search engine (1), and clicks one of the results, which happens to be a malicious page hosted on a compromised server (2). The compromised server forwards the request to a redirection server (3). The redirection server picks an exploit server and redirects the victim to it (4). The exploit server tries to exploit the victim's browser or displays a scareware page (5) to infect the victim through social engineering.

3 Dissecting an SEO Attack

In order to gauge the prevalence of search poisoning attacks, we pick a handful of trendy search terms and issue queries on Google and Bing. Consistent with previous findings, we find that the results to around 36% of the search results contain malicious links (i.e., links that redirect to pages serving malware), with many of the links appearing on the first page of results.

Figure 1 shows, from a legitimate user's perspective, how a victim typically falls prey to an SEO keyword-poisoning attack. The attackers poison popular search terms so that their malicious links show up in the search results for those terms. When the victim uses a search engine to search for such popular terms, some of the results would point to servers controlled by attackers. These are usually legitimate servers that have been compromised by the attackers and used to host SEO pages. Clicking on the search results leads to an SEO page that redirects, after multiple hops, to an exploit server that displays a scareware page. For instance, the scareware page might depict an anti-virus scan with large flashy warnings of multiple infections found on the victim system, scaring the user into downloading and installing an "anti-virus" program. The exploit servers could also try to directly compromise the victim's browser.

To understand exactly how these malicious links end up highly ranked in the search results for popular queries, we pick a few malicious links and examine them closely. Our first observation is that the URLs have similar structure—they all correspond to php files and the search terms being poisoned are present in the URL as arguments to the php file. The SEO page contains content related to the poisoned terms, and also links to URLs of a similar format. These URLs point to SEO pages on other domains that have also been compromised by the same group of attackers. By crawling these links successively till we reach a fixed point, i.e., till we see no more new links, we can identify the entire set of domains involved in a search poisoning attack.

In the rest of this section, we study one particular SEO attack, which started in August 2010, was active for around 10 weeks, and included nearly 37 million SEO pages hosted on over 5,000 compromised servers. Analyzing the php script that generates the SEO page gives us greater insight into the mechanics of this attack. Usually, the source of the php files cannot be obtained directly since accessing the file causes the Web server to execute the php commands and display the output of the execution. In this case, however, we found misconfigured Web servers that did not execute the files, but instead allowed us to download the sources. By examining the source files and log files (the locations of the log files were obtained from the source php file) stored by attackers on the Web server, we get a better understanding of the attack. Note that all the files we examined were publicly accessible without the use of any passwords.

Relying on all of this publicly accessible information, we examine the techniques used by the attackers and identify patterns that help detect other similar attacks. There are three major players in this attack: *compromised Web servers*, *redirection servers*, and *exploit servers*. We discuss each of them in detail next.

3.1 Compromised Web servers

Finding vulnerable servers

The servers were likely compromised through a vulnerability in osCommerce [16], a Web application used to

manage shopping sites. We believe the exploit happened through osCommerce because all of the compromised sites were running the software, and the fake pages were set up under a directory belonging to osCommerce. Additionally, this software has several known vulnerabilities that have remained unpatched for several years, so is a rather easy target for an attacker. Also, the databases associated with shopping sites are likely to store sensitive information such as mailing addresses and credit card details of customers. This offers an additional incentive for attackers to target these sites. We believe that vulnerable servers running osCommerce are discovered using search engines. Attackers craft special queries designed to match the content associated with these Web services and issue these queries to search engines such as Bing or Google to find Web servers running this software [11].

Compromising vulnerable servers

How does the compromise happen? Surprisingly, this is the easiest part of the whole operation. The primary purpose of compromising the site is to store and serve arbitrary files on the server, and to execute commands on the server. With vulnerable installs of osCommerce, this is as easy as going to a specific URL and providing the name of the file to be uploaded. For example, if `www.example.com/store` is the site, then visiting `www.example.com/store/admin/file_manager.php/login.php?action=processuploads` and specifying a filename as a POST variable will upload the corresponding file to the server.

Hosting malicious content

Typically, attackers upload php scripts, which allow them to execute commands on the compromised machine with the privilege of the Web server (e.g., Apache). In many cases, attackers upload a graphical shell or a file manager (also written in php), so that they can easily navigate the files on the server to find sensitive information. The shell includes functions that make it easy for the attackers to perform activities such as a brute-force attack on `/etc/passwd`, listening on a port on the server, or connecting to some remote address.

In our case, the attacker uploads a simple php script, shown in Figure 2. This file is added to the `images/` folder and is named something inconspicuous, so as to not arouse the suspicion of the server administrator. This script allows the attacker to either run a php command, run a system command, or upload a file to the server. A newer version of the script (seen since October 9th, 2010) additionally allows the attacker to change the permissions of a file.

Once this script is in place, the attacker can add files to the server for setting up fake pages that will be indexed by search engines. These files include an html template, a CSS style file, an image that looks like a YouTube player window, and a php script (usually named `page.php`) that puts all the content together and generates an html page using the template. The URLs to the pages set up by the attackers are of the form: `site/images/page.php?page=<keyphrase>`. The set of valid keyphrases is stored in another file (`key.txt`), which is also uploaded by the attackers. Most of the keyphrases in the file are obtained from Google *hot trends* [9] and Bing *related searches*.

```php
<?php
  $e=@$_POST['e'];
  $s=@$_POST['s'];
  if($e) {
     eval($e);
  }
  if($s) {
     system($s);
  }
  if($_FILES['f']['name']!='') {
     move_uploaded_file(
        $_FILES['f']['tmp_name'],
        $_FILES['f']['name']);
  }
?>
```

Figure 2: The `php` script uploaded by the attackers to the compromised server.

In some other attacks, we observe that the attackers make use of *cloaking* techniques [25,27] while delivering malware, i.e., they set up two sets of pages and provided non-malicious pages to search engine bots, while serving malicious pages to victims. In this specific attack, however, the attackers do not use cloaking. Instead, the same page is returned to both search engines and regular users, and the page makes use of javascript and flash to redirect victims to a different page. The redirection is triggered by user actions (mouse movement in this case). The rationale here is that search engine crawlers typically do not generate user actions, so will not know that visitors will be redirected to another URL. Using such flash code for redirection makes detection much harder.

The SEO page

The bulk of the work in creating the SEO page and links is done by the `page.php` script uploaded to the server. This is an obfuscated php script, and like many obfuscated scripts, it uses a series of substitution ciphers followed by an `eval` function to execute the de-obfuscated code. By hooking into the eval function in php, we get the unobfuscated version. The script performs three activities:

1. *Check if search engine:* When the page is requested, the script first checks if the request is from

a search engine crawler. It does this by checking the user-agent string against a list of strings used by search engines. If the request is from a search crawler, the script logs the time of the request, the IP address of the requester, the user-agent string, and the exact URL requested. Since this attack does not use any cloaking, this check seems to be only for logging purposes.

2. *Generate links:* The script loads the html template, and fills in the title and other headings using the keyphrase in the URL. It picks 40 random keyphrases from `key.txt` and generates links to the same server using these keyphrases. It then picks five other keyphrases from `key.txt` and generates links to five other domains (randomly picked from a set of other domains that have also been compromised by the attacker). In all, there are 45 links to similar pages hosted on this and other compromised servers.

3. *Generate content:* Finally, the script also generates content that is relevant to the keyphrase in the URL. It does this with the help of search engines. It queries `google.com` for the keyphrase, and fetches the top 100 results, including the URLs and snippets. It also fetches the top 30 images from `bing.com` for the same keyphrase. The script then picks a random set of 10 URLs (along with associated snippets) and 10 images and merges them to generate the content page.

The content generated for each keyphrase is stored on the server in a cached file, and all subsequent requests for the page are satisfied from the cache, without having to regenerate the content. We believe that the presence of highly relevant information on the page, along with the dense link structures, both within the site and across different compromised sites, result in increasing the pageranks of the Web pages generated by the attacker.

3.2 Redirection servers

The second component in the attack framework is the redirection server, which is responsible for redirecting the victim to a server that actually performs the exploit. Typically, there are one to three additional layers of redirection, before the victim reaches the exploit server. In our case, when a victim visits the compromised site and moves the mouse over the fake YouTube player, he or she gets redirected (using javascript) to another compromised domain, which again performs the redirection. We observed two major domains being used for redirection, and analyzed the working of the redirection server.

When the victim reaches the redirection server, it queries a service named *NailCash* to obtain the URL for redirection. The *NailCash* service is accessed via an http request to `feed2.fancyskirt.com`. The redirection server provides as arguments an API key, a command, and a product ID. In this attack, the redirection server picks randomly among two API keys. It specifies the command as `cmd=getTdsUrl`, and the product ID as `productId=3` (which refers to FakeAV).

During our observation, the URLs requested were only for FakeAV, but it is likely that the same redirection service is used for getting URLs for other types of malware.

The redirection server caches the received URL for 10 minutes, and any requests arriving within those 10 minutes are satisfied from the cache without making a request to `feed2.fancyskirt.com`. Between August 8th, 2010 and October 13th, 2010 the redirection server redirected victims to 453 distinct domains. These domains were very similar in name, and were all hosted on just two $/24$ IP prefixes. One of them was located in Illinois and the other in Amsterdam.

Total	`.in`	`.co.cc`	`.net`	`.com`
191	16	28	73	74

Table 1: Breakdown of exploit server TLDs.

3.3 Exploit servers

Finally, the attacker hosts the actual malicious content on an exploit server. We found 191 different domains being used by the exploit server over time. All the domains were hosted on two IP addresses, one located in Quebec, Canada and the other in Luxembourg. The exploit server does not display the scareware page if the user agent is suspicious (such as a search engine crawler), or if the referrer is missing. It also refuses connections from IP addresses belonging to search engine companies. Most of these domains are either .com, .net, or .co.cc, or .in, and the breakdown is shown in Table 1.

3.4 Results and observations

We present some of the results from our study. Starting with one compromised site, we were able to follow the links to other compromised sites and eventually map the whole network of compromised sites used in this attack. In all, we were able to identify 5400 domains, of which around 5000 were active, and the others were either down or had been cleaned up.

Link structure

Figure 3 shows the number of compromised domains each site links to. On average, each domain linked to 202 other domains, with a median value of 159. In addition, each compromised domain also linked to around 80,000 legitimate domains, since each compromised server had around 8,000 keyphrases, each corresponding to an SEO

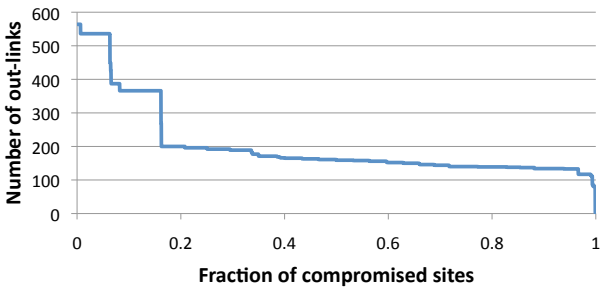

Figure 3: The number of other compromised sites each site links to. The degree distribution indicates a dense linking structure.

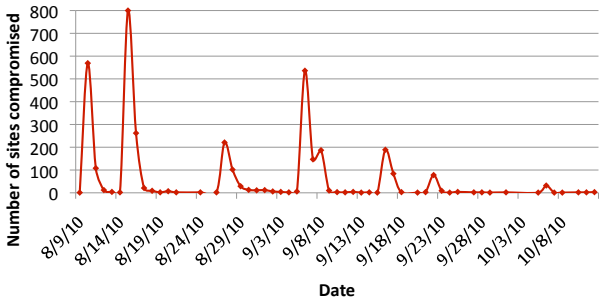

Figure 4: The number of sites compromised by the attackers each day over a period of three months.

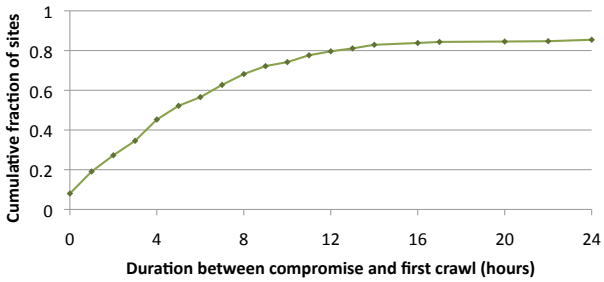

Figure 5: The interval between a site getting compromised and the SEO page getting crawled by a search engine.

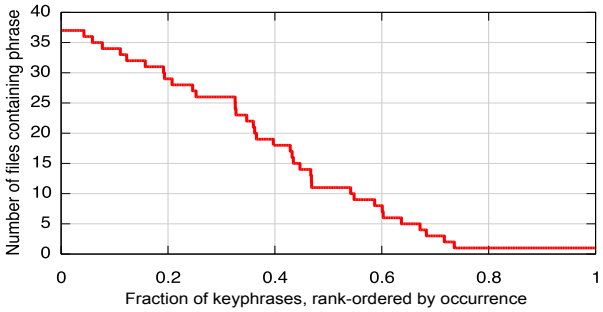

Figure 6: The frequency with which each keyphrase occurs across the compromised sites.

page, linking to 10 different sites obtained from a Google search. The dense link structure helps boost the pageranks of these fake pages in the query results.

Timeline of compromise

Figure 4 shows the number of sites compromised on each day. We define the time of compromise as the time at which the malicious php files were added to the server. This time is obtained from the directory listing on the server. We find the compromise volume to be rather bursty, with most of the servers getting compromised in the initial phase of the attack.

Once the sites are compromised and set up to serve the fake pages, we look at how soon the first visit from search engine crawlers appear.

In Figure 5, we see that almost half of the compromised sites are crawled within four hours of compromise, and nearly 85% of the sites are crawled within a day. This could either be because search engine crawlers are very aggressive at crawling new links, or because the attackers are submitting their sites actively to the search engines through the Webmaster tools associated with each search engine. The dense link structure might also account for the quick crawling time of these pages.

Distribution of keyphrases

Each compromised server sets up an SEO page for each of the keyphrases present in the file. Across all the compromised sites, we found 38 different keyphrase files, with a total of 20,145 distinct keyphrases.

Figure 6 plots the distribution of the keyphrases across the 38 files. The most popular phrases appear in 37 of the 38 files, while nearly 15% of the phrases appear in only a single file. In the median case, each phrase is seen in 11 different files. To check whether Google trends is one of the sources of these keyphrases, we consider all the keywords which were listed as Google trends over a four month period between May 28th, 2010 and September 27th, 2010. Out of the 2,125 distinct trend phrases in this period, 2,018 ($\approx 95\%$) were keyphrases used by the attackers. Exploiting trendy keywords is thus another characteristic of search poisoning attacks to increase the content relevancy.

Traffic from victims

This was a large scale attack exploiting over 5,000 compromised sites, each hosting close to 8,000 SEO pages—for a total of over 40 *million* SEO pages. However, this does not tell us how successful the attack actually was. We would need to take into account what fraction of these pages were indexed by search engines, how many pages showed up in top search results, and how many users clicked on links to these SEO pages. Thus, the measure of success of this SEO campaign would be the

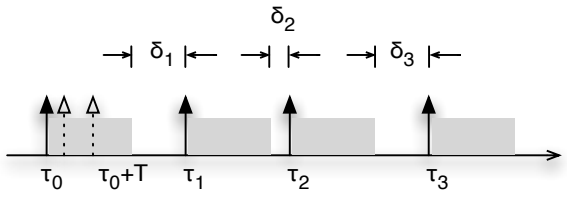

Figure 7: The arrival of requests at the redirection server.

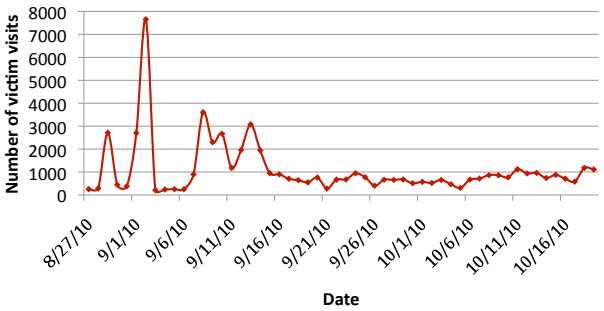

Figure 8: The estimated number of victims redirected to the FakeAV sites on each day.

number of victims who were actually shown the fake anti-virus page.

Unfortunately, the SEO pages on the compromised sites do not log information about who visited each link (unless it is a search engine crawler). However, all the SEO pages cause the user to get redirected to one of two redirection servers. By monitoring the logs on the redirection servers, we estimate the number of visits to the FakeAV pages.

We started monitoring the redirection server on August 27th, 2010, and so missed the first 20 days of the attack. As explained in Section 3.2, the redirection server fetches the redirect-URL from the NailCash service, and each time it does this, it adds an entry to a log file. However, since the URL is cached for 10 minutes, we miss any requests which were satisfied by the cached URL of the exploitation server. Figure 7 illustrates the situation. The solid arrows indicate observed requests (which were written to the log on the redirection server). The grey area denotes the interval when request are served from the cache, and the dotted arrows denote requests which we do not observe because they arrived before the cache entry expired.

In order to estimate the total traffic volume from the observed requests, we make the common assumption that the requests follow a *Poisson* arrival process. This implies that the inter-arrival times are exponentially distributed with mean λ. Since the exponential distribution is memoryless, the time to the next arrival has the same distribution at any instant.

Consider again Figure 7. The first request is observed at time $t = \tau_0$. Since the inter-arrival time is memoryless, the expected time to the next event is the same whether we start our observation at $t = \tau_0$ or at any other t (including $t = \tau_0 + T$). Therefore, we start our observation at $t = \tau_0 + T$, where T is the duration till which a fetched redirect URL is cached, and the time to the next arrival δ_1 is a sample from our exponential distribution. Similarly, $\delta_2, \delta_3, \ldots, \delta_n$ are other samples from this distribution. The mean is then given by:

$$\lambda = \frac{1}{n} \times \sum_{i=1}^{n} \delta_i$$

This is valid only for homogeneous exponential functions, and since Web site visits tend to exhibit diurnal patterns, we split the time into chunks of 2 hours, during which we assume the inter-arrival distribution to be homogeneous. Once we have the mean inter-arrival time λ_i for time interval t_i, we can compute the expected number of visits n_{vis} as:

$$n_{vis} = \sum_{i=1}^{n} \frac{len(t_i)}{\lambda_i}$$

We use this formula to estimate the number of visits to the redirection server, and plot the results in Figure 8. We observe a peak on September 2nd, and then a sudden drop after that for a few days. We believe the drop occurred because the redirection server was added to browser blacklists. On September 7th, the redirection server was moved to another domain, and we start seeing traffic again. The redirection servers stopped working on October 21st, and that marked the end of the SEO campaign. During this period, we estimate the total number of visits to be 60,248, and by extrapolating this number to the start of the campaign (August 7th), we estimate that there were over 81,000 victims who were taken to FakeAV sites. The large number of visits suggests that the attack is quite successful in attracting legitimate user populations.

Multiple Compromises

Perhaps unsurprisingly, we found that many of these vulnerable servers were compromised multiple times by different attackers. We speculate that these were multiple attackers based on the timestamps of when the files were added to the server, and the contents of the files. It is possible that the same attacker uploaded multiple files to the server at different times, but in many cases we see multiple php scripts which offer almost identical functionality, but are slightly different in structure. Also, since we observed the bursty nature of compromises, by looking at timestamps of these uploaded files and clustering the different sites by this timestamp, we can potentially find

groups of sites which were compromised by different attackers at different times.

This observation suggests that attackers share compromised server infrastructure, and thus detecting these sites can effectively help search engines remove a wide class of malicious content.

4 Detection Method

The previous section shows how search-result poisoning attacks are typically performed. In this section, we present our system *deSEO* for automatically detecting such attacks. This task is challenging as it is expensive to test the maliciousness of every link on the Web using content-based approaches. Even if we test only links that contain trendy keywords, it is not straightforward as many SEO pages may look legitimate in response to most requests; they deliver malicious content only when certain environmental requirements are met, e.g., the use of a vulnerable browser, the redirection by search engines, or user actions such as mouse movements. Without careful reverse engineering, it is hard to guess the right environment settings needed to obtain the malicious content on the page.

To automatically detect the SEO links, we revisit the attack we analyzed in the previous section. Our study yields three key observations of why the SEO attack is successful:

1. Generation of pages with relevant content.
2. Targeting multiple popular search keywords to increase coverage.
3. Creating dense link structures to boost pagerank.

Attackers first need to automatically generate pages that look relevant to search engines. In addition, one page alone may not be able to bring them many victims, so attackers often generate many pages to cover a wide range of popular search keywords. To promote these pages to the top of the search results, attackers need to hijack the reputation of compromised servers and create dense link structures to boost pagerank.

We draw on the first two observations when designing our detection method. We do not look at the link structures of Web pages because that would require crawling and downloading *all* pages to extract the cross-link information. In this paper, we show that studying just the structure of URLs works well enough to detect SEO attacks of this type.

Further, we observe that SEO links are often set up on compromised Web servers. These servers usually change their behavior after being hacked: many new links are added, usually with different URL structures from the old URLs. In addition, since attackers control a large number of compromised servers and generate pages using scripts, their URL structures are often very similar across compromised domains. Therefore, we can recognize SEO attacks by looking for newly created pages that share the same structure on different domains. By doing so, we can identify *a group of* compromised servers controlled by the same attacker (or the same SEO campaign), rather than reasoning about *individual* servers.

At a high level, deSEO uses three steps for detection. The first step is to identify suspicious Web sites that exhibit a change in behavior with respect to their own history. In the second step, we derive lexical features for each suspicious Web site and cluster them. In the last step, we perform group analysis to pick out suspicious SEO clusters. Next, we explain these three steps in detail.

4.1 History-based detection

In the first step, deSEO identifies suspicious Web sites that may have been compromised by attackers. SEO pages typically have keywords in the URL because search engines take those into consideration when computing the relevance of pages for a search request [7]. So, we study all URLs that contain keywords. Optionally, we could also focus on URLs that contain popular search keywords because most SEO attacks aim to poison these keywords so as to maximize their impact and reach many users.

While it is common for Web sites to have links that contain keywords, URLs on compromised servers are all newly set up, so their structures are often different from historical URLs from the same domains.

Specifically, for each URL that contains keywords delimited by common separators such as + and −, we extract the URL prefix before the keywords. For example, consider the following URL `http://www.askania-fachmaerkte.de/images/news.php?page=lisa+roberts+gillan`. The keywords in the URL are `lisa roberts gillan` and the URL prefix before the keywords is `http://www.askania-fachmaerkte.de/images/news.php?page=`.

If the corresponding Web site did not have pages starting with the same URL prefix before, we consider the appearance of a new URL prefix as suspicious and further process them in the next step.

4.2 Clustering of suspicious domains

In the second step, deSEO proceeds to cluster URLs so that malicious links from the same SEO campaign will be grouped together, under the assumption that they are generated by the same script.

Similar to previous URL-based approaches for spam detection [13,12], we extract lexical features from URLs.

Empirically, we select the following features:

1. String features: separator between keywords, argument name, filename, subdirectory name before the keywords.

2. Numerical features: number of arguments in the URL, length of arguments, length of filename, length of keywords.

3. Bag of words: keywords.

In our previous URL example, the separator between keywords is "+", the argument name is `page`, the filename is `news.php`, the directory before the keywords is `images`, the number of arguments in the URL is one, the length of arguments is four, the length of filename is nine, and the bag of words is {`lisa, roberts, gillan`}

As most malicious URLs created on the same domain have similar structure, we aggregate URL features together by domain names and study the similarities across domains. Note that we consider sub-domains separately. For example, `abcd.blogspot.com` is considered a separate domain because it is possible for a subdomain to get compromised rather than the entire domain `blogspot.com`. When aggregating for string features, we take the feature value that covers the most URLs in the domain; for numerical features, we take the median; and for bags of words, we take the union of bags.

In contrast to previous work that use URLs for spam detection, where a binary classification of URL is sufficient [13, 12], our goal is to cluster URLs. We adopt the widely used K-means++ method [2]. Initially, we select K centroids that are distant from each other. Next we apply the K-means algorithm to compute K clusters. We select and output clusters that are tight, i.e., having low residual sum of squares (the squared distance of each data point from the cluster centroid). For the remaining data points, we iteratively apply the K-means algorithm until no more big clusters (with at least 10 domains) can be selected.

Note that neither the computation of distances between data points nor the calculation of the cluster centroid is straightforward because we have many features with some of them being non-numerical values. We normalize feature dimensions so that distances fall into a weighted high-dimensional space, with the values of each dimension ranging from 0 to 1. For string features, identical values have a distance of 0 and the distance is set to 1 otherwise. For numerical features, we define the distance as the difference in numerical values, normalized by the maximum value seen in the dataset. For bags of words features, the distance between two bags of words $A = a_1, a_2, ..., a_n$ and $B = b_1, b_2, ..., b_m$ is defined as $\frac{\|A \cap B\|}{\|A \cup B\|}$. When picking a weight for each dimension, we give a higher weight (the value 2) to string features as it is relatively infrequent for different URLs to have identical string features. For all other dimensions, we give an equal weight of 1.

When computing centroids, we adopt the same method we use to aggregate URL features into domain features, treating all URLs in a cluster as if they were from the same domain. If we find a cluster with the residual error of squares normalized by the cluster size lower than the preset threshold, we output the cluster. Empirically, we find both the weight selection and the threshold selection are not sensitive to the results as most malicious clusters are very tight in distance and stand out easily.

4.3 Group analysis

Finally, we perform group analysis to pick compromised domain groups and filter legitimate groups. In the previous steps, we leverage the fact that compromised sites change behavior after the compromise and their link structures are similar. In this step, we leverage another important observation, namely that SEO links in one campaign share a similar page structure (not just the URL structure).

One way to measure the similarity of two Web page structures is to compare their parsed HTML tree structure [21]. This approach is heavy-weight because we need to implement a complete HTML page parser, derive the tree representations, and perform tree difference computations. For simplicity, we focus on simpler features that are effective at characterizing pages. For instance, we simply use the number of URLs in each page, and we find this feature works well empirically.

We sample N (set to 100) pages from each group and crawl these pages. Then we extract the number of URLs per page and build a histogram. Figure 9 plots the histogram of the number of URLs of a legitimate group, while Figure 10 plots the histogram for a malicious group. We can clearly see that the legitimate group has a diverse number of URLs. But the malicious one has very similar pages with almost identical number of URLs per page. The small fraction of zero link pages are caused by pages that no longer exist (possibly corresponding to compromised Web servers that have since been cleaned up).

We normalize each histogram and compute peaks in the normalized histograms. If the histogram has high peak values, we output the group as a suspicious SEO group and manually check the group. Although here we still use manual investigation to pick out the final groups, the amount of work is actually small. We show in Section 5 that deSEO outputs less than 20 groups. Therefore, a human expert only needs to check several sample URLs in each group, rather than reasoning about millions

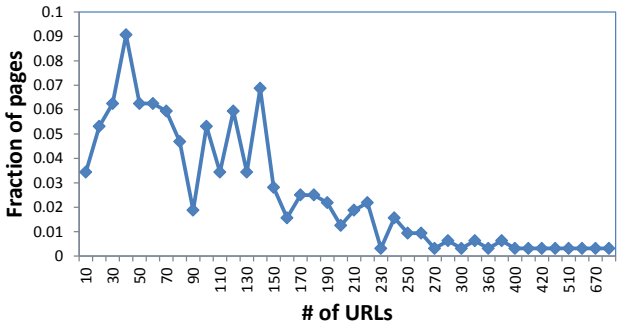

Figure 9: An example legitimate group that has diverse distribution of number of URLs in each Web page.

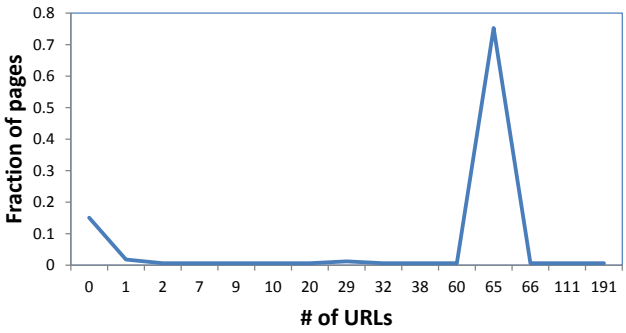

Figure 10: An example malicious group that has a similar number of URLs in each Web page.

of URLs that contain keywords one by one.

Finally, for each malicious group, deSEO outputs regular expression signatures using the signature generation system AutoRE [28]. Since URLs within a group have similar features, most groups output only one signature. We apply the derived signatures to large search engines and are able to capture a broad set of attacks appearing in search engine results (see details in Section 6.3).

5 Results

In this section, we describe our datasets, which consist of large sets of Web URLs, search engine query logs, snapshot of Web content, and trendy keywords. Using these datasets, we evaluate the effectiveness of deSEO in identifying malicious groups of URLs corresponding to different SEO attacks.

5.1 Dataset

We collect three sampled sets of URLs from Bing. These URLs are sampled from all URLs that the crawler saw during the months of June 2010, September 2010, and January 2011. Each sampled set of URLs contains over a hundred billion URLs. We use the June URLs as a historical snapshot and apply deSEO to September and January URL sets.

The second dataset we use is a sampled search query log from September 2010 that contains over 1 billion query requests. It records information about each query such as query terms, clicks, query IP address, cookie, and user agent. Because of privacy concerns, cookies and user agents are anonymized by hashing. In addition, when we look at IP addresses in the log, we focus on studying the IP addresses of compromised Web servers, rather than individual normal users.

The trendy keywords we use are obtained from Google Trends [9]. We collect daily Google Trends keywords from May 28th, 2010 to February 3rd, 2011. Each day has 20 popular search terms.

5.2 Detection results

5.2.1 History-based detection

We apply the history-based detection to the URLs of September and January. Since we have over a hundred billion URLs, to reduce the processing overhead, we first filter out the top 10,000 Alexa [1] Web sites as we believe those servers are relatively well managed and have a lower chance of getting compromised. Later, after we derive regular expression patterns, we could apply them to URLs corresponding to these Web sites to detect malicious ones, if any, hosted by these servers.

Month	With trendy keyword		With new structure	
	Domains	URLs	Domains	URLs
Sept 10	428,430	1,481,766	136,387	366,767
Jan 11	512,617	3,255,140	211,225	1,102,878

Table 2: History-based URL filtering.

We extract all URLs on remaining domains that contain trendy keywords. Table 2 shows the results. In September, over 1 million URLs have trendy keywords, but in Jan the number jumps to 3 million, showing the potential increase of SEO attacks. We next choose URLs with new URL prefixes by comparing the URL prefixes of September 2010 and January 2011 to those of June 2010. For URLs that contain new prefixes, we select them and pass them to the next step. This step removes about two thirds of the URLs.

5.2.2 Clustering results

We extract the domain features as described in Section 4.2 and apply the K-means++ algorithm to cluster these domains. We vary the value of K and obtain similar results since we apply the K-means++ algorithm iteratively. Table 3 shows the results of K=100. (The third and fourth columns are explained below.) For both months, the clustering algorithm outputs hundreds of groups.

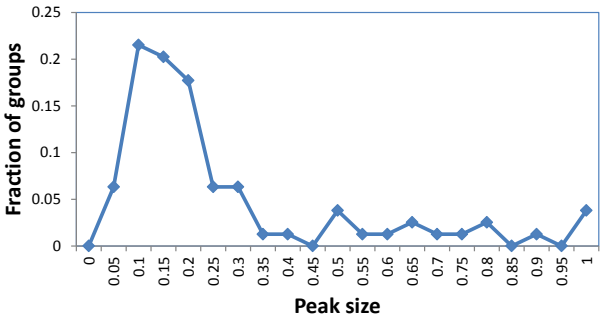

Figure 11: The distribution of peak values: percentage of pages sharing the same number of URLs within a group.

5.2.3 Group analysis

As we have grouped similar Web sites into a small number of groups, we use group similarity to distinguish legitimate groups from malicious ones. We use the URL features mentioned in Section 4.3 to filter out obvious false-positive groups.

Figure 11 shows the distribution of the peak value among all groups. We can see that there are a small number of groups that have high peak values. But most groups have small peak values as their pages are diverse. We pick a threshold for the peak value of 0.45, which filters most legitimate groups, as shown in the third column of Table 3. After filtering, less than 20 groups remain, and we manually go through these groups to pick out malicious ones.

Month	Number of groups		
	Total	Above threshold	Malicious
Sept 10	290	14	9
Jan 11	272	16	11

Table 3: Clustering and group analysis results.

In total, we find 9 malicious groups from the September data and 11 groups from the January data. The regular expressions derived from two datasets mostly overlap. This shows that there are a relatively small number of SEO campaigns, and that they are long-lasting. Hence, capturing one signature can be useful to capture many compromised sites over time. In total, we capture 957 unique compromised domains and 15,482 malicious URLs in our sampled datasets.

Figure 12 shows a few derived regular expression samples. These include expressions that match the URLs of compromised servers that we study in Section 3, but also a number of new ones. Note that some of the regular expressions may look generic, e.g., `*/index.php/?w{4,5}=(w+(+w+)+)$`, which matches malicious URLs like: `http://www.kantana.com/2009/index.php/?bqfb=justin+bieber+breaks+neck`. At first glance, one might think `index.php` followed by words would match many legitimate URLs, but it turns out that it is rare to have "`/?`" in between. Further, the word `bqfb` makes it even clearer that this is an automatically generated URL.

6 Attack Analysis

In this section, we leverage the results produced by deSEO to gain more insights into SEO attacks. First, we study a new attack found by deSEO, which has a different link structure than the one we detect in Section 3. Second, we study the search engine queries originating from the IP addresses of compromised servers, as SEO toolkits often query the search engines to generate SEO pages. Finally we apply the derived regular expressions to live search results to detect a broad set of attacks.

6.1 Study of new attack

By examining deSEO's captured malicious groups, we find another SEO attack that uses a different methodology for setting up SEO pages, boosting their page ranks, and polluting the search index. We believe that this SEO campaign is probably orchestrated by a different group of attackers. We now characterize the differences between this attack and the attack that we initially studied.

6.1.1 Link structure

This attack makes use of two sets of servers—one set that hosts SEO pages that redirect to an exploit server, and a second set of pointer pages that link to only these SEO pages.

We find 120 pointer pages, all of which are hosted on hacked Wordpress [24] blogs. Further analysis shows that these are older versions of Wordpress that have vulnerabilities, and attackers use one of these vulnerabilities to modify the `xmlrpc.php` files that are included in the Wordpress installation by default. Each pointer page contains 500 links to SEO pages hosted on 12 different domains. The pointer pages are dynamic in that the set of links contained in a page changes each time the page is visited, and the set of the 12 domains also changes on a daily basis.

In all, we find SEO pages hosted on 976 domains. Similar to the previous attack, the SEO pages contain content relevant to the poisoned terms and redirect users to the exploit server. However, new to this attack, the SEO pages did not link to each other. Instead, they relied on incoming links from the pointer pages to boost their pageranks, as well as to populate the search engine index with new SEO pages. However, in addition, the SEO pages started linking to each other starting in January 2011. This change suggests that the attackers are

Regex	Examples
.*\/images\/\w+(-\w+)+\.html	http://usedcarsdotcom.com.au/images/eddie-fisher.html
.*\/image\/page\.php\?page=\w+(\+\w+)+	http://www.rawstrokes.com/cart/images/page.php?page=justin+bieber+hates+korea&check=dd35923778116c82bc9c5b102ea9e260
.*\/robots.txt\/\?showc=\w+(\+\w+)+$	http://www.soundsonshellac.com/robots.txt/?showc=tour+de+france+stage+3
.*\/xmlrpc\.php\/\?showc=\w+(\+\w+)+$	http://randomlyinsaneadventures.com/xmlrpc.php/?showc=sec+media+days+2010
.*\/images\/watch\/index\.php\?q=(\w+(\+\w+)+)$	http://www.po-kwong.com/product/images/watch/index.php?q=justin+moore
.*\/[a-z]{4,5}.php\?[a-z]{3,5}=\w+(\+\w+)+$	http://eleishamiller.com/nofsj.php?page=steven+pieper
*\/[a-z]{3,7}\.php\?[a-z]{1,7}=(\w+(%20\w+)+)$	http://vott500.com/ufdvq.php?go=caliphate%20definition
.*\/index\.php\/\?\w{4,5}=(\w+(\+\w+)+)$	http://www.kantana.com/2009/index.php/?bqfb=justin+bieber+breaks+neck

Figure 12: Examples of derived regular expressions.

constantly trying to improve the ranking of their pages using different strategies.

6.1.2 Use of cloaking

This attack makes use of cloaking, both for the pointer pages and the SEO pages. When a pointer page is accessed by a legitimate user, the original page content is displayed, but if the page is accessed by a search-engine crawler (identified by the user-agent string), then a page containing links to the SEO pages is displayed.

The SEO pages behave differently depending on how they are accessed. When accessed by a search engine crawler, the page displayed is optimized for the poisoned keywords. When a regular user accesses the page, he/she is redirected to the exploit server, provided the referrer field matches a known search engine and the user-agent field indicates a Windows machine. In all other cases, the SEO page redirects to a benign page.

6.1.3 Redirection and exploit infrastructure

This attack makes use of a completely different set of redirection and exploit servers, though the FakeAV page displayed at the end is almost identical. In comparison with the previous attack, these SEO pages go through an extra level of redirection for reaching the exploit server. We find a total of 485 exploit domains, hosted on two sets of IP address in the US (in Texas and New Jersey).

6.2 Queries from compromised servers

We check whether we see queries from the compromised servers captured by deSEO using the Bing search log. Queries from Web servers can be viewed as a signal of potential compromise, as they could indicate search engine scraping activities in order to generate content for SEO pages. Less than 5% of the top 500 Alexa Web sites ever submitted queries during the month of September 2010, while 46% of the compromised servers did. Note that this does not necessarily mean the remaining ones do not issue queries to search engines. They could have been inactive during that month, or could have chosen to use other search engines.

The queries from legitimate sites are mostly infrequent (less than one per day). These may be generated by human administrators, who are logged in on the Web servers. Only around 1% of legitimate sites generated a large number of queries. These queries went through the affiliate program that partners with the search engine company to provide search results.

Queries from the IPs of compromised servers are more frequent than those of legitimate sites. In addition, queries from the same group have similar behavior. Often, they present the same user-agent string, e.g., "`Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US; rv:1.9.2) Gecko/20100115 Firefox/3.6`". Relying on this user-agent string, together with trendy keywords, we detect other IP addresses that share the same pattern. Accurately determining the compromised domains hosted on these IP addresses is challenging, though, because compromised servers are usually small Web servers hosted on hosting infrastructures. It is common to have many domains (sometimes tens of thousands) sharing the same IP address. Therefore, seeing bad activities from an IP address is not sufficient to pinpoint the exact compromised server.

Besides trendy keyword queries, we also identify a number of other malicious queries from these compromised servers. For example, there are queries of the form of `site:`$< hosting\_site >$. This query returns all the pages from a particular site. What is interesting is that the site specified in the query is also hosted on the same IP address that issued the query. Such queries are seen when a site is compromised, and the attackers try to de-

	60 trendy keywords		60 attacker poisoned keywords	
	# of matched searches	# of matched URLs	# of matched searches	# of matched URLs
Google	16	39	27	124
Bing	0	0	1	1

Table 4: Matching Google and Bing search results using derived regular expressions.

termine which pages to inject code into; they typically pick the most popular pages, i.e. the ones that show up high in the search results.

6.3 Matching Google and Bing queries

We apply our derived regular expressions of Section 5.2.3 to the Google and Bing search engines. We use two sets of query terms. The first set is a set of 60 trendy keywords obtained from Google Trends (February 1st to February 3rd, 2011). The second set is a set of 60 keywords poisoned by attackers (but not in Google Trends), which were randomly selected from the keywords appearing in captured malicious URLs. For each keyword, we manually perform Web search and then extract the top 100 results returned from Google and Bing. We use only 60 search terms because the search queries are issued manually—automated queries and screen-scraping are against the terms of use, and the search results obtained using the search APIs are not consistent with the results obtained through a browser. (While we do not know why the API results differ from the browser-based search results, we speculate that the API search results are not as fresh, so contain older and more well-established links.)

For a total of 120 keywords, 36% of them yield at least one malicious link in the top 100 results (which are spread over ten pages). Table 4 shows the detailed results. Not surprisingly, attackers are even more successful in poisoning non-trendy keywords that they select (45% match rate). This is because fewer Web pages may match these keywords and hence it can be easier for malicious links to appear among the top search results. Their distribution, i.e., how high in the search results these links are displayed, is shown in Figure 13. We can see that the malicious links are spread over all of the top 10 pages. Similar to previous reports [19], we find Bing top search results contain relatively fewer malicious links. (Experiments were conducted in February 2011; the search results of both Google and Bing have been improved since then.)

We manually verified all the matches and did not find false positives. All of the matches are generated by only two regular expressions—the first and the seventh in Figure 12.

We run all matching URLs through Firefox using the Google Safebrowsing API and Internet Explorer (using its internal blacklist), and none of them were blocked by either browser at the time of the experiment. This result indicates that deSEO is able to capture live attacks that have not yet been reported.

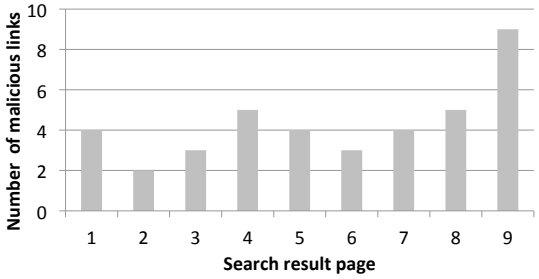

Figure 13: The number of malicious links found in different pages of the search results for 60 popular keywords.

7 Discussion and Conclusion

In this paper, we study a large-scale, live search-result poisoning attack that leverages SEO techniques. Based on our observations, we develop a system called deSEO that automatically detects additional malicious SEO campaigns. By deriving URL signatures from our results and applying them to both Google and Bing, we find 36% of searches yield links to malicious pages among their top results. Our paper appears to be the first to present a systematic study of search-result poisoning attacks and how to detect them.

Attackers may wish to evade deSEO detection by not embedding keywords in URLs. However, this approach reduces the chance of getting SEO links to the top search results, because keywords in URLs appear to be an important feature for relevance computation. Also, it may reduce the chance of clicks by end users, as URLs with keywords look more relevant to users who search using these keywords. Attackers may also wish to diversify the SEO link structures so that they look different across different domains. Our history-based detection will still pick such SEO links as long as their URL structures appear different than those used previously by the same domains. To further detect the diversified SEO links as a group, we could alternatively adopt content-based solutions by comparing their page similarity [21], possibly on virtual machines [22].

In our study, we find that attackers usually put up a large number of new pages after compromising a Web site, which is often relatively inactive before compromise. Therefore, search engines could give a lower rank to new pages on previously inactive sites. Search engines could also consider dense link structures to identify SEO attacks, if they are willing to crawl most of the malicious pages and if they can afford to perform offline analysis. In addition, we notice that the contents of SEO pages are mostly irrelevant to the compromised site's homepage, and sometimes even the language is different. Therefore, semantic-based approaches are also promising avenues for further investigation.

8 Acknowledgements

We thank Úlfar Erlingsson, Qifa Ke, and Marc Najork for their valuable advice. We are grateful to Fritz Behr, David Felstead, Nancy Jacobs, Santhosh Kodipaka, Liefu Liu, Cheng Niu, Christian Seifert, David Soukal, Walter Sun, and Zijian Zheng for providing us with data and feedback on the paper. The investigation and analysis of SEO attacks was partly supported by a Cisco Fellowship and the National Science Foundation under Grants CNS-0831540 and CNS-1040663.

References

[1] Alexa's Top 1 million Web sites. http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

[2] D. Arthur and S. Vassilvitskii. K-means++: the advantages of careful seeding. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA, 2007.

[3] V. Aubuchon. Google Ranking Factors - SEO Checklist. http://www.vaughns-1-pagers.com/internet/google-ranking-factors.htm.

[4] C. Castillo, D. Donato, A. Gionis, V. Murdock, and F. Silvestri. Know your neighbors: Web spam detection using the Web topology. In *Proceedings of the 30th International ACM Conference on Research and Development in Information Retrieval*, SIGIR, 2007.

[5] D. Fetterly, M. Manasse, and M. Najork. Spam, damn spam, and statistics: using statistical analysis to locate spam Web pages. In *Proceedings of the 7th International Workshop on the Web and Databases*, WebDB, 2004.

[6] R. Fishkin and J. Pollard. Search engine ranking factors, 2009. http://www.seomoz.org/article/search-ranking-factors.

[7] Googleguide. How Google works. http://www.googleguide.com/google_works.html/.

[8] Google search engine optimization starter guide. http://www.google.com/webmasters/docs/search-engine-optimization-starter-guide.pdf.

[9] Google trends. http://www.google.com/trends.

[10] Hotvideo pages: analysis of a hijacked site. http://research.zscaler.com/2010/09/hot-video-pages-analysis-of-hijacked.html.

[11] J. P. John, F. Yu, Y. Xie, M. Abadi, and A. Krishnamurthy. Searching the Searchers with SearchAudit. In *Usenix Security Symposium*, 2010.

[12] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Beyond blacklists: Learning to detect malicious Web sites from suspicious urls. In *Proceedings of the SIGKDD Conference*, 2009.

[13] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker. Identifying suspicious urls: An application of large-scale online learning. In *Proceedings of the International Conference on Machine Learning*, ICML, 2009.

[14] A. Moshchuk, T. Bragin, S. D. Gribble, and H. M. Levy. A crawler-based study of spyware on the Web. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS, 2006.

[15] A. Ntoulas, M. Najork, M. Manasse, and D. Fetterly. Detecting spam Web pages through content analysis. In *Proceedings of the International Conference on World Wide Web*, WWW, 2006.

[16] osCommerce, Open Source Online Shop E-Commerce Solutions. http://www.oscommerce.com.

[17] L. Page, S. Brin, R. Motwani, and T. Winograd. The pagerank citation ranking: Bringing order to the Web. 1998. http://www.scientificcommons.org/42893894.

[18] M. A. Rajab, L. Ballard, P. Mavrommatis, N. Provos, and X. Zhao. The nocebo effect on the Web: an analysis of fake anti-virus distribution. In *Proceedings of the 3rd USENIX Conference on Large-scale Exploits and Emergent Threats: Botnets, Spyware, Worms, and More*, LEET, 2010.

[19] Spam SEO trends & statistics. http://research.zscaler.com/2010/07/spam-seo-trends-statistics-part-ii.html.

[20] Twitter trending topics. http://twitter.com/trendingtopics.

[21] T. Urvoy, E. Chauveau, P. Filoche, and T. Lavergne. Tracking Web spam with html style similarities. *ACM Transactions on the Web*, February 2008.

[22] Y.-M. Wang, D. Beck, X. Jiang, R. Roussev, C. Verbowski, S. Chen, and S. King. Automated Web patrol with strider honeymonkeys. In *Proceedings of the Network and Distributed System Security Symposium*, NDSS, 2006.

[23] Websense 2010 threat report. http://www.websense.com/content/threat-report-2010-introduction.aspx.

[24] WordPress. http://wordpress.org.

[25] B. Wu and B. D. Davison. Cloaking and redirection: A preliminary study. In *Adversarial Information Retrieval on the Web*, AIRWeb, 2005.

[26] B. Wu and B. D. Davison. Identifying link farm spam pages. In *Special Interest Tracks and Posters of the International Conference on World Wide Web*, WWW, 2005.

[27] B. Wu and B. D. Davison. Detecting semantic cloaking on the Web. In *Proceedings of International Conference on World Wide Web*, WWW, 2006.

[28] Y. Xie, F. Yu, K. Achan, R. Panigrahy, G. Hulten, and I. Osipkov. Spamming botnets: Signatures and characteristics. In *SIGCOMM*, 2008.

[29] Yahoo! Search Content Quality Guidelines. `http://help.yahoo.com/l/us/yahoo/search/basics/basics-18.html`.