PriveonLabs Research

Cisco Security Agent Protection Series:

Using Inline Frames for Malware Delivery
(IFRAME Element)

Zach Brewer
Security Consultant

Overview

Previous Priveon Labs research documents discussed specific browser vulnerabilities including the MS RDS.Dataspace (CVE-2006-003 / MS06-014) and XMLHTTP ActiveX (CVE-2006-5745 / MS06-071) vulnerabilities. These attack vectors are often found in the wild occurring in multiple instances on one or more sites. Often, common web-based attack vectors include the use of HTML IFrame tags in order to distribute malware to unsuspecting users. In this document, we will attempt to explain the use of IFrame elements in malware delivery including real-world examples. We will also outline the role of Cisco Security Agent in preventing IFrame delivered malware within a layered endpoint security strategy.

Inline Frames Explained

The IFRAME HTML element is simply used to include inline frames within the context of a web page. In its most basic form, the IFrame element displays inline content that exists on another webpage. The referenced webpage may or may not be on the same server as the referencing page as seen in Figure 2.

Figure 1: Basic Inline Frame Code Example

```
<iframe src="somepage.html" width=100%"></iframe>
```

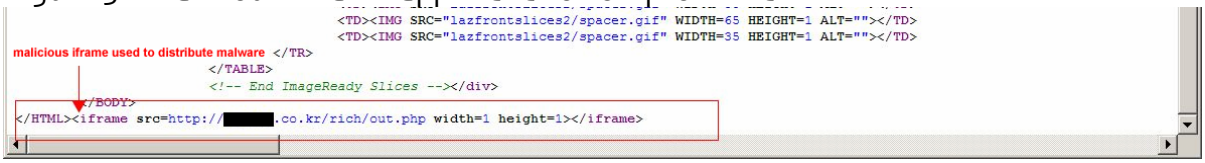
Figure 2: Basic Browser Rendering of an Inline Frame Tag

For more information on the IFrame HTML element, please see http://www.w3schools.com/tags/tag_iframe.asp.

IFRAME AS AN ATTACK VECTOR

Inline frames can be used as an attack vector when a website references another page that hosts malicious content through the IFrame tag. It is important to remember that the page referenced may or may not be on the same server. As a result, it is often possible for someone to add malicious content to an otherwise non-malicious site by simply adding a single line of HTML code to one or more pages on the compromised server. The example shown in figure 3 was discovered during the course of a Priveon Labs incident response investigation.

Figure 3: Malicious IFrame appended to compromised site



In most cases, the IFrame tag is hidden by simply setting both the width and height attributes to 1 pixel. More than one inline frame may be used to reference several vulnerabilities in order to increase an attacker's chance of successful exploitation. Hidden inline frames may also include various techniques to obfuscate the code through methods such as URL encoding. The JavaScript URL encoding technique increases the difficulty of malicious HTML code detection. Figure 4 shows an IFrame that is obfuscated using the URL encoding method. Figure 5 shows the same code without URL encoding. Both encoding methods are rendered the same in most browsers. .

Figure 4: IFrame elements encoded using JavaScript and URL encoding

```
document.write(unescape("%3C%69%66%72%61%6D%65%20%73%72%63%3D%22%73
%6F%6D%65%73%69%74%65%31%2E%63%6F%6D%22%3E%3"));
```

Figure 5: Decoded IFrame Elements referenced in figure 4

```
<iframe src="somesite1.com"></iframe>
<iframe src="somesite2.com"></iframe>
<iframe src="somesite3.com"></iframe>
```

Exploit Process Overview

On 12/29/2006, the SANS Internet Storm Center (ISC) disclosed a website that was being used to distribute malware through inline frames. As referenced in the SANS ISC Diary entry, the exploit was spammed as postcard.exe. This malware is referenced as win32.exe when an exploit results in code execution. As expected, the site utilizes multiple IFrame references in order to exploit multiple vulnerabilities including MS06-014 (RDS Dataspace Vulnerability), MS06-057 (WebViewFolderIcon), and MS06-055, (VML vulnerability), and MS06-071 (XML Core Services). The site also uses URL encoding in order to obfuscate the referenced code. (For more information on the aforementioned individual vulnerabilities, please see previous Priveon Labs CSA Protection Series documents at www.priveon.com/research)

The code discussed in the SANS article represents an all too common method of malware delivery in the wild. As a result, Priveon Labs obtained and modified the code in order to test "real-world" code and exploits against a default CSA ruleset.

PoC Exploit Testing Overview

Testing Environment

The Priveon test lab consisted of one windows XP system that was deemed to be vulnerable to at least one of the vulnerabilities referenced in the IFrame code. Additionally, the lab contained one simulated web server hosting modified exploit code as well as the win32.exe malware. The vulnerable Windows XP system was placed in the "Desktops – All Types" CSA group. The simulated victim host was placed into test mode in order to show rules that would have been triggered during each stage of malware delivery in a real world exploit process.

Cisco Security Agent Version Tested:

- CSA 5.1.0.69 (Default CSA Policies as shown in Figure 6)

Figure 6: Protected Host CSA Policies from v5.1.0.69

Group Name	Version	Description	Policies
<All Windows>		Auto-enrollment group for Windows hosts	2 policies

Policy Name	Version	Description	Rule Modules
Application Classification	5.1 r69	Base policy for behavioral classification of applications.	3 modules
Operating System - Base Permissions - Windows	5.1 r69	Basic permissions for Windows OS	2 modules

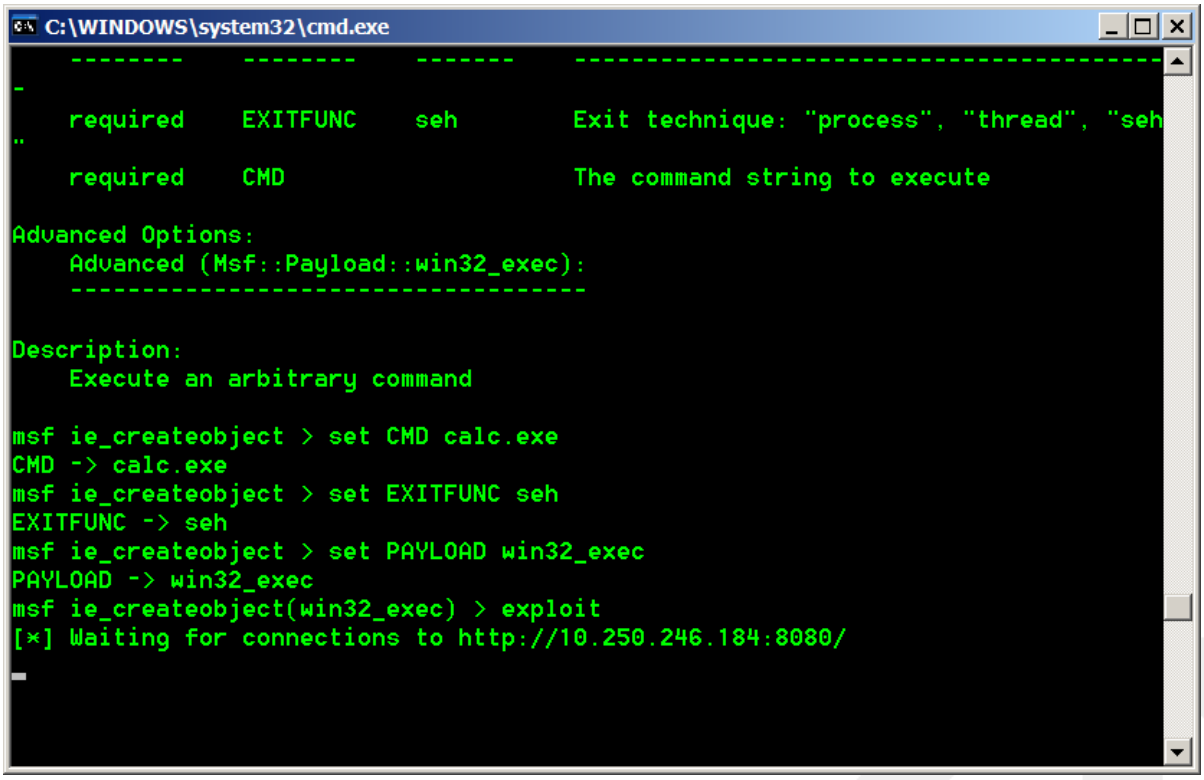
Desktops - All types	5.1 r69	Default group for systems that install the Desktop agent kit	9 policies

Policy Name	Version	Description	Rule Modules
Agent UI control	5.1 r69	Policy which governs Agent User Interface	1 module
Document Security - Windows	5.1 r69	Policy to protect user documents	1 module
Email Client - Basic Security - Windows	5.1 r69	Basic application enforcement policy for email client software.	3 modules
General application - Basic Security - Windows	5.1 r69	Basic, Application independent security policy for Windows	3 modules
Installation Applications - Windows	5.1 r69	Software Installers for Windows	4 modules
IP Stack - Internal Network Security	5.1 r69	Policy for protecting the IP Stack on internal systems	1 module
Network Personal Firewall	5.1 r69	Control network access and provide some end user access controls.	1 module
Operating System - Base Protection - Windows	5.1 r69	Basic protection for Windows OS	6 modules
Virus Scanner - Windows	5.1 r69	Application enforcement policy for virus scanner software.	1 module

Exploit PoC Testing Results – Unprotected System

The first step in re-creating the delivery mechanism in a quarantined environment was to verify vulnerability to one of the exploits used by the malicious site. This process was verified by using the MetaSploit framework. Since one of the exploits in question was MS06-14, the MetaSploit ie_createobject exploit module was used to verify vulnerability by using a simple CMD_Exec payload to invoke calc.exe.

Figure 7: MetaSploit waiting for connection to simulated web server



Once the victim host was deemed to be vulnerable to at least one of the exploits referenced in the Inline Frames, the PoC testing continued with the Cisco Security Agent installed.

CSA Protected Exploit PoC Testing Results

With the Cisco Security Agent installed in testmode on the simulated target, Internet Explorer was directed to the simulated attacking web server. Each of the resulting rules triggered demonstrated the various layers of protection provided by the default CSA desktop ruleset. It should be noted, however, that the entire attack would have been successfully prevented with the enforcement of the initial Application Control rule. Had our system not been in test mode, this Application Control rule would have been the only rule observed. The other rules observed in Figure 8 resulted from the behavior of the win32.exe delivered malware rather than the exploit process itself. CSA would have successfully prevented the initial behavior as well as certain behavioral aspects of the delivered malware.

Figure 8: CSA log of simulated malware delivery through IFrame HTML elements and IE vulnerabilities

Summary of Results

CSA successfully prevented exploitation and execution of the win32.exe malware through the use of Application Control and other rules. Each of the vulnerabilities referenced in the IFrame HTML elements from the malicious website would have been contained through CSA's defense-in-depth approach to rule enforcement on a simulated victim system. The ability to prevent against unknown threats and new malware delivery mechanisms on an unpatched system is why CSA is a necessary element of a layered strategy to endpoint security.

References

- http://isc.sans.org/diary.html?storyid=1983
- http://www.milworm.com

Where to Go for More Information:

Custom Research Documents
Exploit Reverse Engineering/Forensics
Security Implementation
Real-World Training
Managed Services

Available @ www.Priveon.com