CFR Watering Hole Attack Details

By by Darien Kindlund

Published: 2012-12-28 · Archived: 2026-04-05 20:20:16 UTC

Threat Research

[**Updated on December 30, 2012**] On December 27, we received reports that the Council on Foreign Relations (CFR) website was compromised and hosting malicious content on or around 2:00 PM EST on Wednesday, December 26. Through our Malware Protection Cloud, we can confirm that the website was compromised

at that time, but we can also confirm that the **CFR website was also hosting the malicious content as early as Friday, December 21—right before a major U.S. holiday**.

We can also confirm that the malicious content hosted on the website does appear to use Adobe Flash to generate a heap spray attack against Internet Explorer version 8.0 (fully patched), which was the source of the zero-day vulnerability. We have chosen not to release the technical details of this exploit, as Microsoft is still investigating the vulnerability at this time.

In the meantime, the initial JavaScript hosting the exploit has some interesting features. To start, it appears the JavaScript only served the exploit to browsers whose **operating system** language was either English (U.S.), Chinese (China), Chinese (Taiwan), Japanese, Korean, or Russian:

```
var h=navigator.systemLanguage.toLowerCase();
if(h!="zh-cn" && h!="en-us" && h!="zh-tw" && h!="ja" && h!="ru" && h!="ko")
{
 location.href="about:blank";
}
```

Also, the exploit used browser cookies to ensure that the exploit is only delivered once for every user:

```
var num=DisplayInfo();
if(num >1)
{
 location.href="about:blank";
}
```

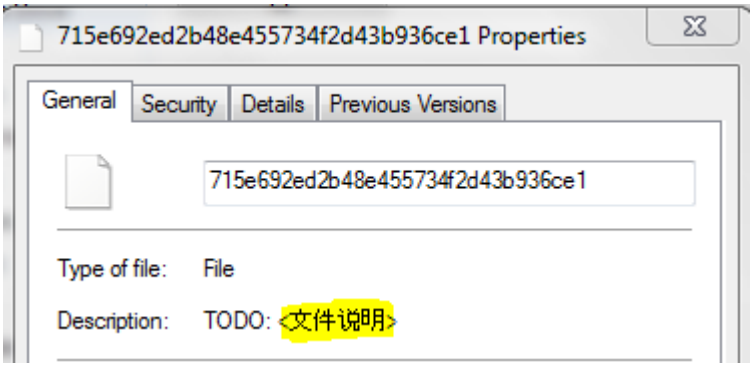
where DisplayInfo() essentially tracked when the page was last visited through browser cookies:

```
function DisplayInfo()
{
   var expdate = new Date();
   var visit;
```

```
    expdate.setTime(expdate.getTime() +  (24 * 60 * 60 * 1000*7 ));
    if(!(visit = GetCookie("visit")))
    visit = 0;
    visit++;
    SetCookie("visit", visit, expdate, "/", null, false);
    return visit;
}
```

Once those initial checks passed, the JavaScript proceeded to load a flash file "today.swf", which ultimately triggered a heap spray in Internet Explorer in order to complete the compromise of the endpoint.

Once the browser is exploited, **Prior to downloading the exploit**, the browser appears to download “xsainfo.jpg” **into the browser cache (aka "drive-by cache attack")**, which is the dropper encoded using single-byte XOR (key: 0x83, ignoring 0x00 and 0x83 bytes). **Once the exploit succeeds, the JPG file is decoded as a DLL, which is written to the %TEMP% folder as "flowertep.jpg", the sample** (MD5: 1e90bd550fa8b288764dd3b9f90425f8) (MD5: **715e692ed2b48e455734f2d43b936ce1**) appears to contain debugging information in the metadata of the file:



The simplified Chinese <文件说明> translates to <File Description>. It looks like the malware author included additional metadata in the file, referencing "test_gaga.dll" as the internal name of this sample.

[**Update: December 30, 2012**]

Additionally, another binary "shiape.exe" (MD5: **a2e119106c38e09d2202e2a33e64adc9**) is initially dropped to the %TEMP% folder and executed, which appears to perform code injection activity against the standard IE "iexplore.exe" process and installs itself as "%PROGRAMFILES%\Common Files\DirectDB.exe", using the HKLM\SOFTWARE\STS\"nck" registry key to maintain state information. We can also confirm Jamie Blasco's (AlienVault) findings, where the malware registers "DirectDB.exe" through the active setup registry key HKLM\SOFTWARE\Microsoft\Active Setup\Installed

Components\, so that this executable starts once per user upon subsequent login. However, while AlienVault's IOC mentions a process handle name of "&!#@&", **we have actually found the "shiape.exe" process generates a mutex of \BaseNamedObjects\&!#@& upon execution, as well**.

As of December 29, Microsoft released a security advisory (2794220), which provides initial details of the zero-day vulnerability. Additionally, CVE-2012-4792 has been assigned to track this vulnerability over time.

Simultaneously, Metasploit developers have analyzed the vulnerability and Eric Romang released details of the CDwnBindInfo Proof of Concept Vulnerability demonstration for independent verification.

Lastly, Cristian Craioveanu and Jonathan Ness with MSRC Engineering posted further technical details on MS Technet about ways to mitigate CVE-2012-4792, in the meantime.

One interesting side-note that MSRC mentioned is:

> *On systems where the vulnerability is not present, this Javascript snippet will have the side effect of initiating an HTTP GET request with the encoded heap spray address in it. A network log or proxy log would reveal the following HTTP requests:*
>
> ```
> GET /exploit.html HTTP/1.1
>
>
>
> GET /%E0%B4%8C%E1%82%ABhttps://www.example.com HTTP/1.1
> ```
>
> *As you can see, the value 0x10AB0C0D is encoded in UTF8 and sent as part of the HTTP request. The real-world exploits do not use example.com and the heap spray address will vary depending on targeted OS platform and exploit mechanism but if you see encoded memory addresses in your proxy log, you should investigate to determine whether your organization has been targeted.*

Yet, in the wild, we have seen cases where a **successfully compromised endpoint** generates an equivalent encoded pattern:

```
GET /js/js/%E0%B4%8C%E1%88%92https://www.google.com/settings/account HTTP/1.1\r\n
[truncated] Accept: image/gif, image/jpeg, image/pjpeg, image/pjpeg, application/x-shockwave-flash, appli
Referer: http://www.cfr.org/js/js/news.html\r\n
Accept-Language: en-us\r\n
User-Agent: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; .NET CLR 2.0.50727; .NET CLR
Accept-Encoding: gzip, deflate\r\n
Host: www.cfr.org\r\n
Connection: Keep-Alive\r\n
Cookie: visit=1; NREUM=s=1356553990828&r=56794&p=78217\r\n
\r\n
[Full request URI: http://www.cfr.org/js/js/%E0%B4%8C%E1%88%92https://www.google.com/settings/account]
```

... and **still beacons** (via HTTP POST) to the initial command and control infrastructure at [REDACTED].**yourtrap.com**. **Therefore, we encourage security operations analysts to fully investigate endpoints that generate the encoded heap spray network traffic, rather than simply assuming those systems are not compromised.**

We will continue to update this blog with further details, as we discover new information about this attack.
