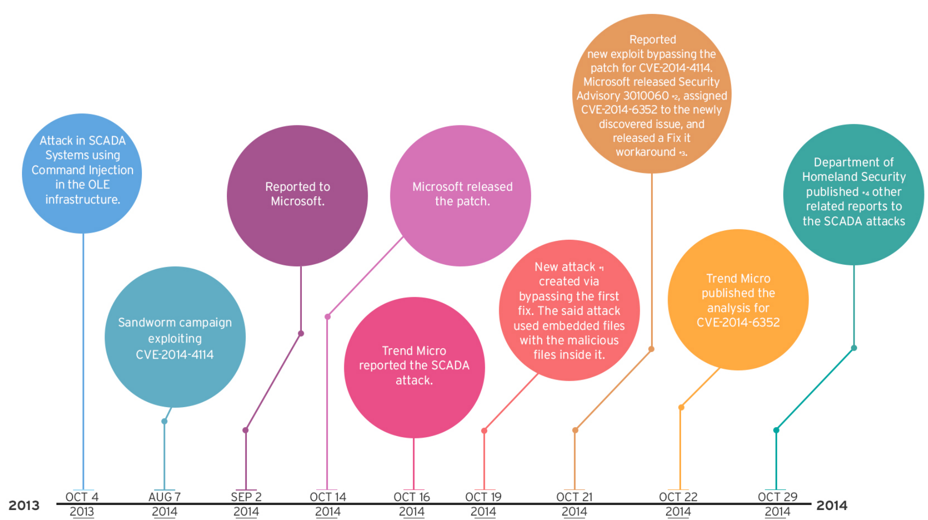
Timeline of Sandworm Attacks | Security Intelligence Blog

William Gamazo Sanchez (Vulnerability Research)

Share Tweet

The Sandworm vulnerability, also known as CVE-2014-4114, is an interesting vulnerability for two reasons. For one, it is related to the timing of the vulnerability life cycle. In this blog post, we will tackle vulnerability analysis, and user awareness on what actions to take when they are under attack. Note that all dates and times discussed here are based on publicly available information and in the internal metadata of the sample files. Here's a timeline:



Click image to enlarge

*1: New CVE-2014-4114 Attacks Seen One Week After Fix
*2: https://technet.microsoft.com/library/security/3010060
*3: https://support.microsoft.com/kb/3010060
*4: https://ics-cert.us-cert.gov/alerts/ICS-ALERT-14-281-01A

CVE-2014-4114 is also related to the OLE design by itself. We can classify it as a Command Injection in the OLE infrastructure. This area is sufficiently complex and its hard to evaluate the scope of the attack surface; this caused the release of an incomplete fix and the release of CVE-2014-6352. This is because an attacker can control two external variables to invoke different paths inside the affected component *package.dll*. The variables are: OLE Verbs and Embedded File Type.

Vulnerability time cycle

Looking at the timelines is always helpful to understand and correlate major events. Sandworm became known to the public when iSIGHT released a blog entry on October 14 discussing the vulnerability and how it was being used in targeted attacks. It was fixed on the same day as part of the scheduled Patch Tuesday release, in MS14-060. A week later, on October 21, it was disclosed that under certain circumstances the patch could be bypassed, resulting in Microsoft Security Advisory 3010060 and published workarounds.

What was in the patches? We found that they contained a new version of the file *packager.dll*. The following image shows the Windows properties of the file:

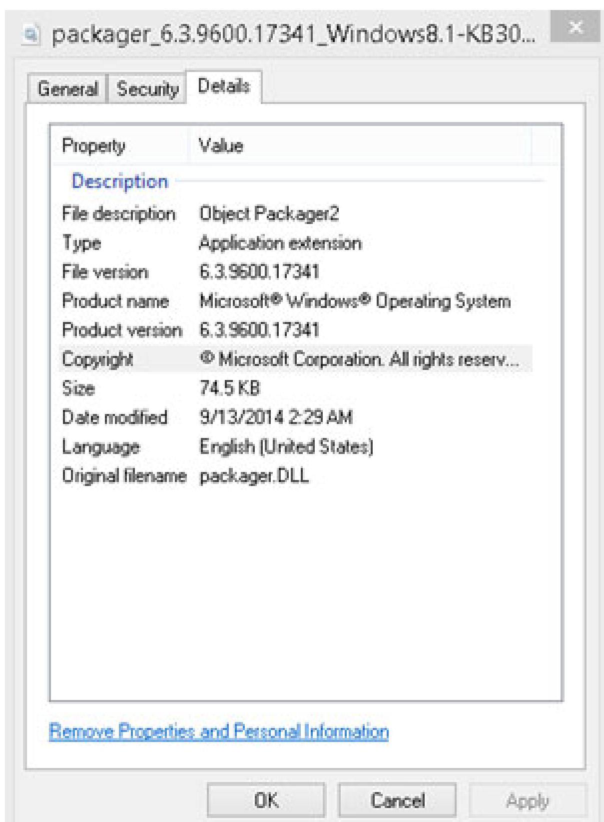
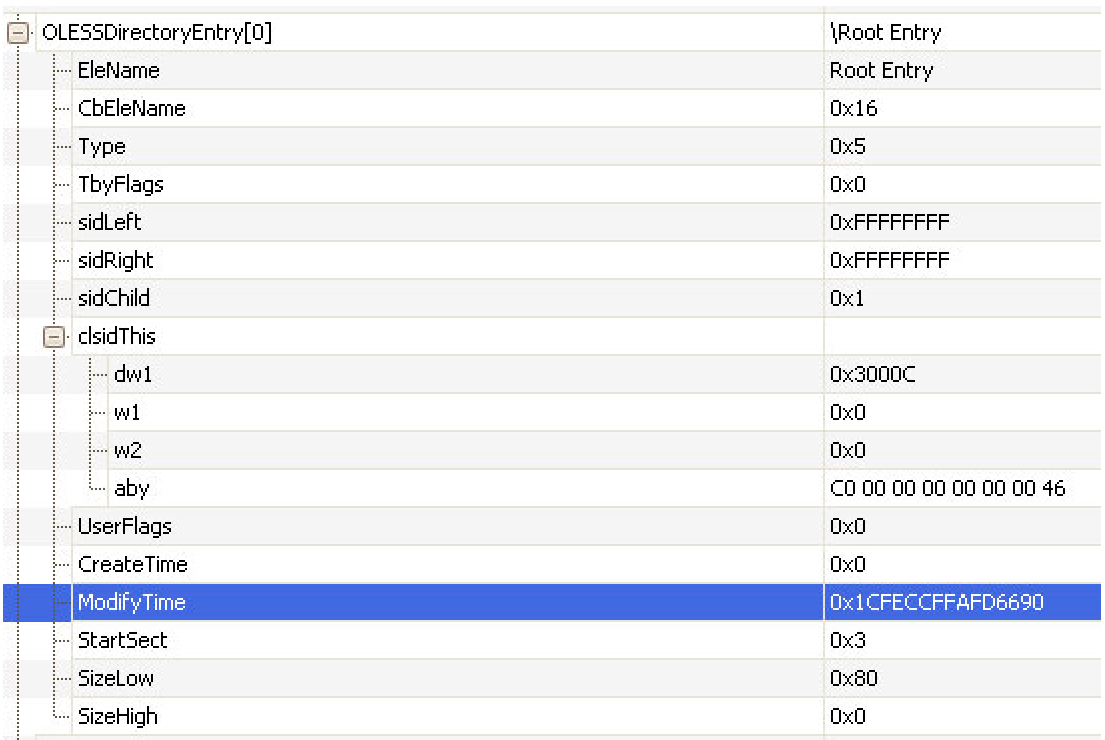


Figure 1. Package.dll updated version (6.3.9600.17341) Windows file properties

This file was created on September 13 – which is reasonable, since iSIGHT first spotted this attack on September 3. Other security vendors indicate they reported this flaw to Microsoft on September 2.

The email campaign of Sandworm (or BlackEnergy) that targeted this vulnerability took place from August 13 onwards, as reported in various articles. These emails used a PPSX attachment with two embedded files. These embedded files contain an internal property informing the modification and created time. The following image shows this property:



OLESSDirectoryEntry[0]	\Root Entry
EleName	Root Entry
CbEleName	0x16
Type	0x5
TbyFlags	0x0
sidLeft	0xFFFFFFFF
sidRight	0xFFFFFFFF
sidChild	0x1
clsidThis	
dw1	0x3000C
w1	0x0
w2	0x0
aby	C0 00 00 00 00 00 00 46
UserFlags	0x0
CreateTime	0x0
ModifyTime	0x1CFECCFFAFD6690
StartSect	0x3
SizeLow	0x80
SizeHigh	0x0

Figure 2. OLE Compound tree structure. Here we can see the ModifyTime is highlighted.

A known file (SHA256 hash: 70b8d220469c8071029795d32ea91829f683e3fbbaa8b978a31a0974daee8aaf) used in this campaign is detected by Trend Micro as TROJ_MDLOAD.PGTY. The embedded files *oleObject1* and *OleObject2* have the modified date/time of 8/7/2014 1:15:59 PM. Following the timeline until here, this would seem like a valid and logical date. On October 16, 2014, Trend Micro reported that the same type of attack is being used to exploit SCADA systems. The said attack employed the same technique – Command Injection in the OLE infrastructure – and used the same file origin. In this case two OLE files were used: *devlist.cim* and *config.bak*. Both files were created on 10/4/2013.

There are several samples in VirusTotal related to this campaign. Some of these samples are directly related to the attacks, while others are simple modification to the attacks done by analysts. Extracting the attack IPs from all the samples we can get the following list:

- \\10[.]0[.]0[.]34\public\slide1.gif
- \\10[.]0[.]0[.]34\public\slide1.inf
- \\10[.]0[.]0[.]27\share\xxx.inf
- \\10[.]0[.]0[.]27\share\xxx.gif
- \\10[.]80[.]65[.]87\impct\losslides.gif
- \\216[.]66[.]74[.]22\/root/smb4k/teamths\ths.inf
- \\216[.]66[.]74[.]22\/root/smb4k/teamths\ths.gif
- \\210[.]209[.]86[.]152\p\z\slides.inf
- \\210[.]209[.]86[.]152\p\z\slides.gif
- \\185[.]29[.]8[.]212\share\sliiides.inf
- \\185[.]29[.]8[.]212\share\sliiides.exe
- \\121[.]166[.]55[.]120\file\lint.inf
- \\121[.]166[.]55[.]120\file\head.gif
- \\121[.]166[.]55[.]12\file\head.gif
- \\192[.]168[.]10[.]10\shared\msf\XrHI.inf
- \\192[.]168[.]10[.]10\shared\msf\XrHI.inf
- \\192[.]168[.]10[.]10\shared\msf\TBSZ.gif
- \\192[.]168[.]1[.]122\Support\xxx.gif
- \\192[.]168[.]1[.]11\share\xxx.inf
- \\192[.]168[.]1[.]11\share\xxx.gif
- \\192[.]168[.]187[.]147\xpl\calc.gif
- \\192[.]168[.]15[.]4\rdb\blah.gif
- \\192[.]168[.]58[.]95\rdb\test.gif
- \\192[.]168[.]58[.]95\rdb\test.inf
- \\192[.]157[.]198[.]1\public\word.gif
- \\118[.]99[.]13[.]236\docs\partyhis.gif
- \\37[.]59[.]5[.]18\11\test.gif
- \\109[.]163[.]233[.]151\public\aaaa.gif
- \\109[.]163[.]233[.]151\public\aaaa.inf
- \\94[.]185[.]85[.]122\public\slide1.inf (This is from the sample mentioned before)
- \\94[.]185[.]85[.]122\public\slide1.gif (This is from the sample mentioned before)
- \\94[.]185[.]85[.]122\public\default.txt (This is the sample attacking SCADA Systems)

First patch and second attack

In this blog post we analyzed how the attacker can control the OLE Verb to execute the file once the PPSX is run. However, another interesting part of the attack is how the attacker control the file type to bypass the Mark on the Web (MOTW) and avoid the alert message in Windows showing the file as untrusted. The user can control the file type using the CLSID in the OLE compound document. The said property is under /Root Entry of the embedded object. The following image shows one example. In this case, the embedded type is 0x22602.

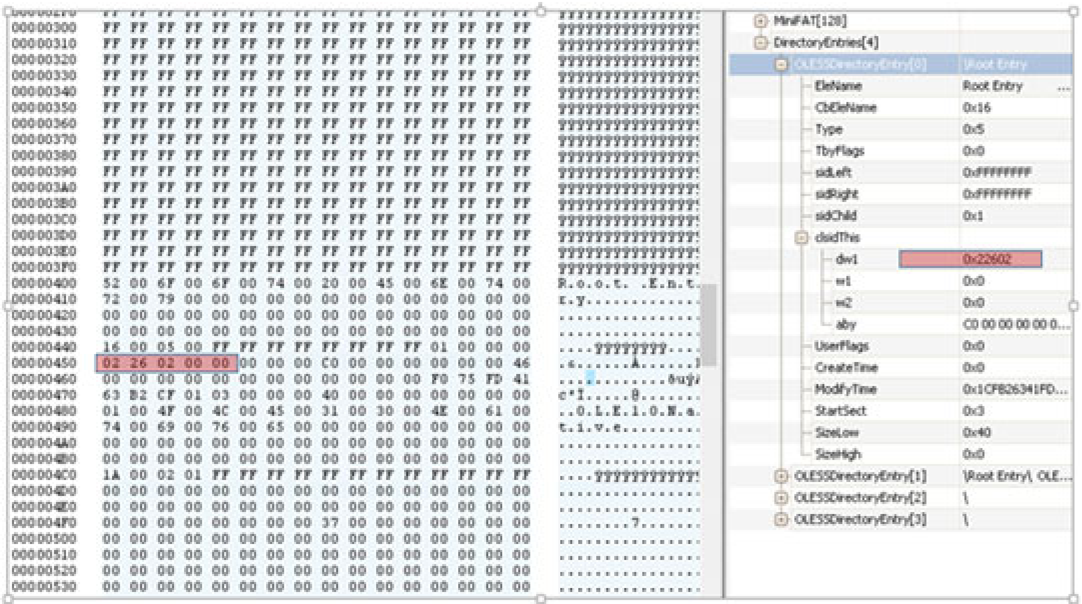


Figure 3. OLE RootEntry property CLSID. The first value is the embedded type(0x22602).

When *package.dll* is processing embedded files, the actual operation or extraction of the file depends on the file type. There are several type of files. The following image shows a big picture on how this works.

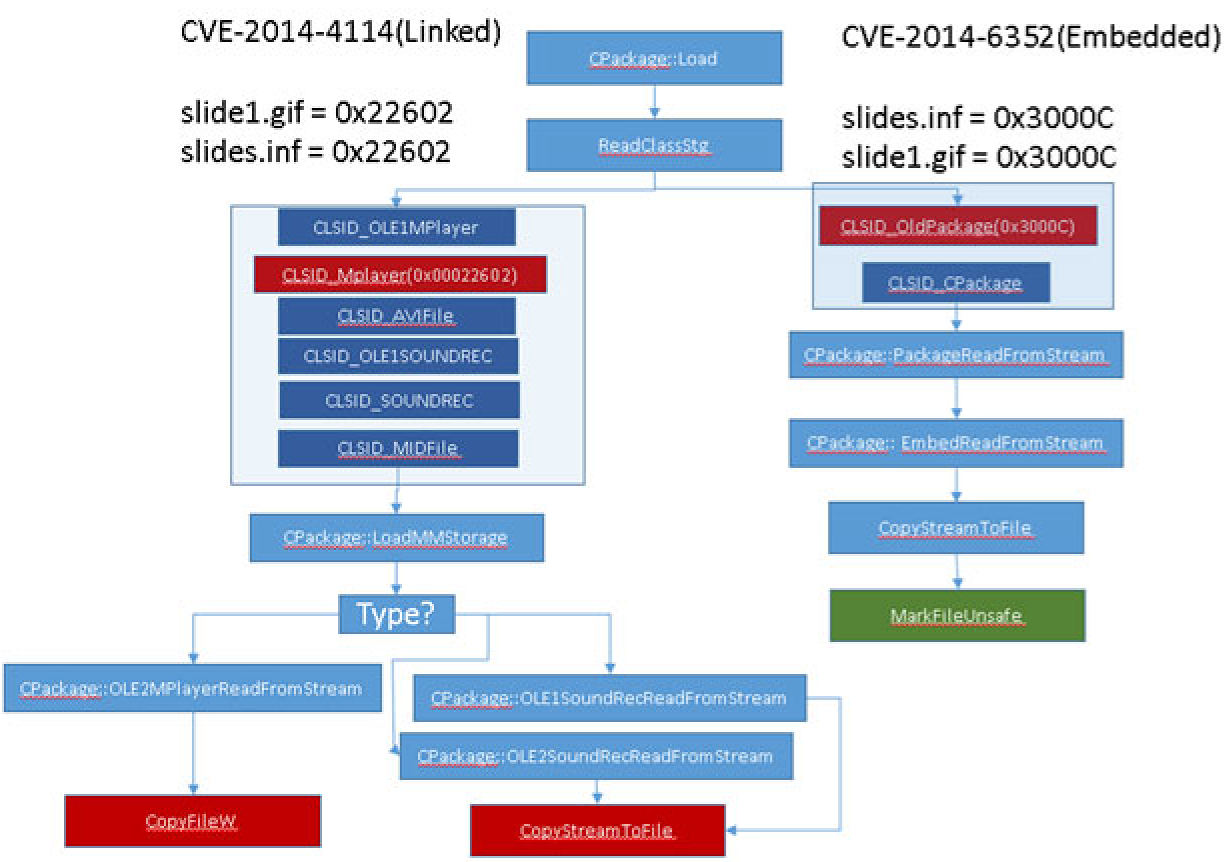


Figure 4. Call paths inside package.dll.

The attack for CVE-2014-4114 used *0x22602* as file type. This allows the attacker to bypass the MOTW protection. The OLE infrastructure will call *CPackage::Load* for each embedded file included in the PPSX file. This method calls *ReadClassStg* to get the embedded file type, which is *0x22602* in both cases. This type is MPlayer. Next, *CPackage::Load* will call *LoadMMSStorage*. The method *LoadMMSStorage* calls *OLE2MPlayerReadFromStream* or *OLE1SoundReadFromStream* depending on the OLE file type returned by *ReadClassStg,* which is *MPlayer* in this case.

The problem is that methods call to *CopyFileW* or *CopyStreamToFile* both will result in creating the temporary file without MOTW. This is because the first patch from Microsoft changed the "XXReadFromStream" methods to call *MarkFileUnsave*. After the first patch the protection looks like the following screenshot:

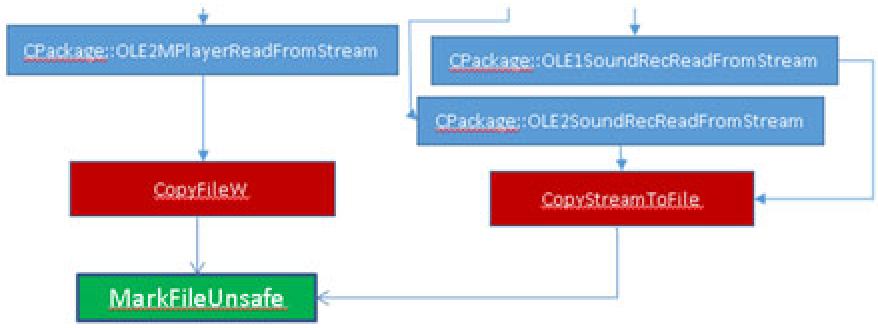


Figure 5. Protection using MOTW after patch.

Note that the automatic execution using specific OLE Verb was not patched. The patch only added MOTW protection for these methods.

For the attack related to CVE-2014-6352, the protection MOTW is not bypassed, as seen in the image before, but the execution will take place showing the following message to the user:

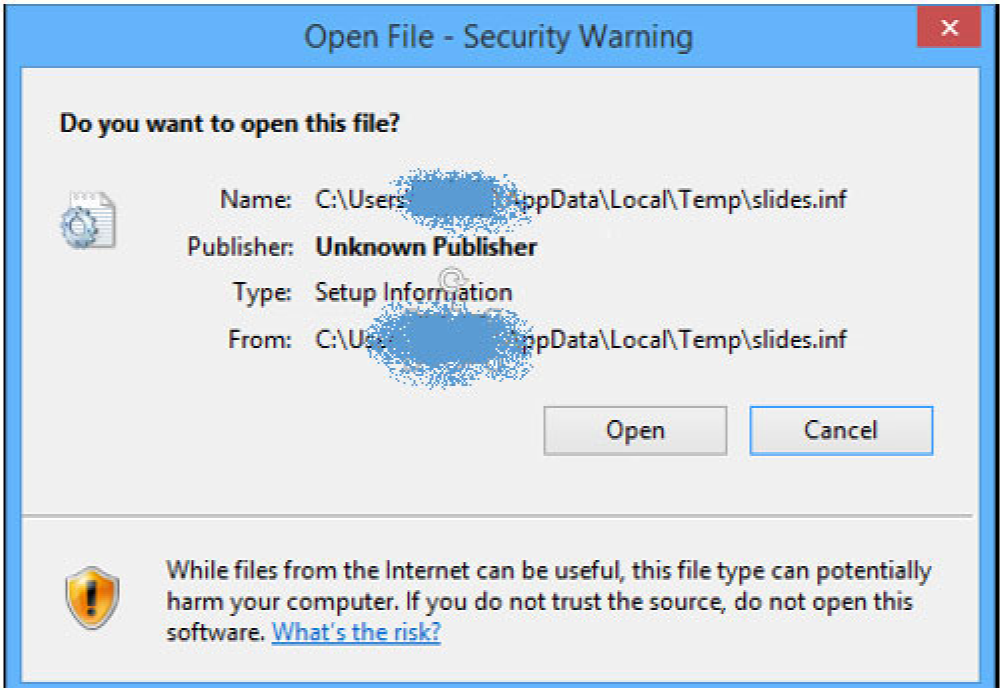


Figure 6. . Pop-up message alerting the user when the file is protected with MOTW.

The MOTW protection will create one NTFS stream to the created file that Windows will use to check to launch the warning message. The created NTFS stream is seen in the following image:

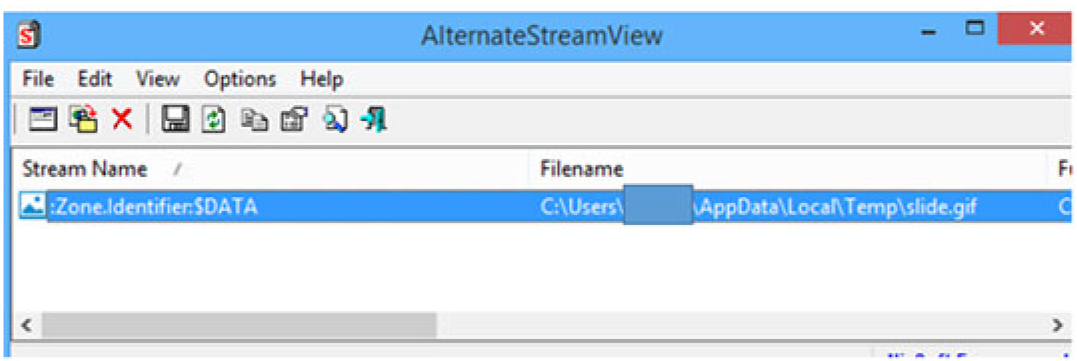


Figure 7. NTFS stream of a file with MOTW activated.

Conclusion

The attack technique for Command Injection in the OLE Infrastructure has been around since at least October 2013. If the attack happens in a system where the patch MS14-060 has been applied, the user will see the warning message shown in Figure 6.

Trend Micro secures users from this threat via detecting the exploit and malware payload via the Smart Protection Network. Trend Micro Deep Security and Office Scan with the Intrusion Defense Firewall (IDF) plugin protect user systems from threats that may leverage this vulnerability via the following DPI rules:

- 1006290 – Microsoft Windows OLE Remote Code Execution Vulnerability (CVE-2014-4114)
- 1006291 Microsoft Windows OLE Remote Code Execution Vulnerability (CVE-2014-4114) – 1

Users are strongly advised to patch their systems once Microsoft releases their security update for this. In addition, it is recommended for users and employees not to open PowerPoint files from unknown sources as this may possibly lead to a series of malware infection.

With additional insights from Pawan Kinger