#BadWinmail:
The "Enterprise Killer" Attack Vector in Microsoft Outlook

by Haifei Li (haifei.van@hotmail.com), December, 2015
current version 1.1, always check the latest version of this paper here

Introduction

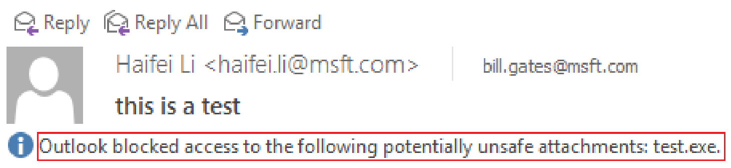
Microsoft Outlook, a part of the Microsoft Office suit, has become one of the most popular applications in today's computing world, especially for the enterprise environment. Enterprise employees use Outlook to exchange emails everyday as well as manage various information such as schedules, meeting invitations, etc. For more information please visit its wiki page.

Acknowledgements: The author would like to thank Xiaoning Li for peer-reviewing the report. Thanks to the Microsoft Security Response Center and the Office team for fixing the issue within 1.5 months. Logo credit: the Pangu Team.

Security Mitigations/Enhancements on Outlook

Since Outlook is such a critical application, Microsoft has implemented various security mitigations/ enhancements to ensure Outlook is safe to use, these include:

- Some file types, such as those bringing direct code executions, are blocked automatically. For example, a .exe file will be blocked automatically without further confirmation from the user, as the following figure shows:

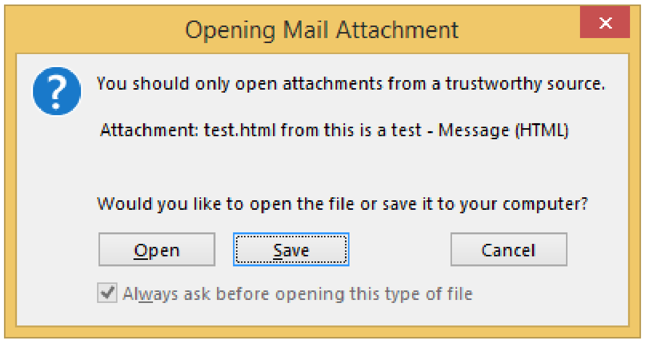


Haifei Li <haifei.li@msft.com>   bill.gates@msft.com

this is a test

Outlook blocked access to the following potentially unsafe attachments: test.exe.

- For those file types that may have potential risks, Outlook offers a warning dialog to the user when the user tries to open the attachment. Following is the warning dialog when trying to open a .html file. Users are not allowed to open such attachment directly.


Opening Mail Attachment

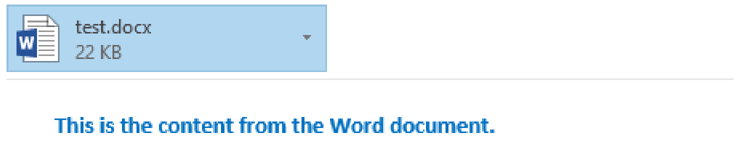
You should only open attachments from a trustworthy source.

Attachment: test.html from this is a test - Message (HTML)

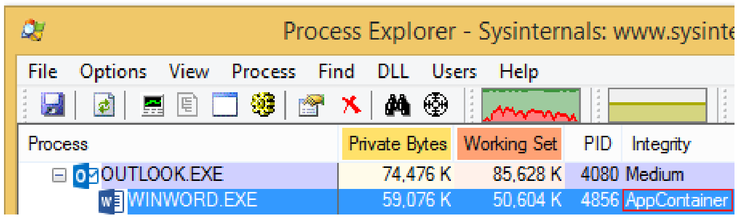
Would you like to open the file or save it to your computer?

[✓] Always ask before opening this type of file

- For Office documents, such as Word, PowerPoint or Excel files, users can either open the attachment by double-clicking on the attachment, or even "previewing" the attachment by simply single-clicking on the attachment icon. Following figure shows the user is previewing the content of a Word document on Outlook 2016.


test.docx
22 KB

This is the content from the Word document.


Process Explorer - Sysinternals: www.sysinte

Process	Private Bytes	Working Set	PID	Integrity
OUTLOOK.EXE	74,476 K	85,628 K	4080	Medium
WINWORD.EXE	59,076 K	50,604 K	4856	AppContainer

Regardless whether it's opened via previewing or actual opening - the document will be rendered in the "Office Sandbox", this is also known as the Protected View feature of Office. According to this MWR Labs research, the sandbox is pretty strong, which makes end users highly immune from Office-based threats delivered via Outlook.

However, in-depth research has showed that there are critical security problems in Outlook, which may be leveraged to bypass those forementioned mitigations. Specially, the author has discovered a novel attack vector in Outlook, which allows anonymous attacker to take control of a computer via just an email. Following we are going to discuss about the details.

The OLE Mechanism

As we know, the Object Linking and Embedding (OLE) technology is well used in the Office Word, Excel, PowerPoint, as well as the WordPad application. For more details about the OLE feature in Office documents, please check out the research entitled "Attacking Interoperability: An OLE Edition" presented at Black Hat USA 2015.

However, previous research only discussed OLE objects embedded in various Office (or RTF) documents, but not for Outlook or emails. The author has found that OLE is also supported in Outlook, which poses a pretty serious security problem.

The "Enterprise Killer" Attack Vector: OLE via TNEF

The Transport Neutral Encapsulation Format (TNEF), is a Microsoft-invented email format supported by Outlook (the author suspects it's only supported by Outlook). For more details please refer to this wiki page.

A "TNEF" email's original content may look like the following:

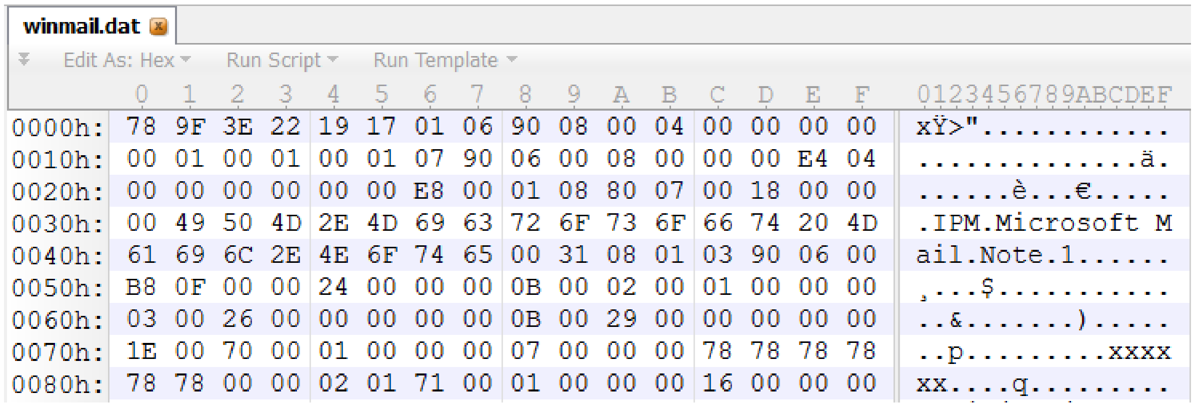
```
------=_NextPart_000_0048_01D106A0.9042DC90
Content-Type: application/ms-tnef;
            name="winmail.dat"
Content-Transfer-Encoding: base64
Content-Disposition: attachment;
            filename="winmail.dat"

eJ8+IhkXAQaQCAAEAAAAAAABAAEAAQeQBgAIAAAA5AQAAAAAAADoAAEIgAcAGAAAAElQTS5NaWNy
b3NvZnQgTWFpbC5Ob3RlADEIAQOQBgC4DwAAJAAAAAsAAgABAAAAAwAmAAAAAAAALACkAAAAAAB4A
```

As shown above, the value of the "*Content-Type*" field is set to "*application/ms-tnef*", and the filename is usually "*winmail.dat*". The "content" is actually a file (after base64 decoding) following the "TNEF" file format, the TNEF file format is well described by Microsoft here.

P.S.: the author named the attack vector as "BadWinmail" because of the special filename "winmail.dat" in the "TNEF" email.

As described in the TNEF specification, when the value of the "*PidTagAttachMethod*" is set to *ATTACH_OLE (6)*, the "attachment file" (which is another file contained in the winmail.dat file) will be rendered as an OLE object, the same description can also be found online at the MSDN site.

A sample winmail.dat file may look like the following:



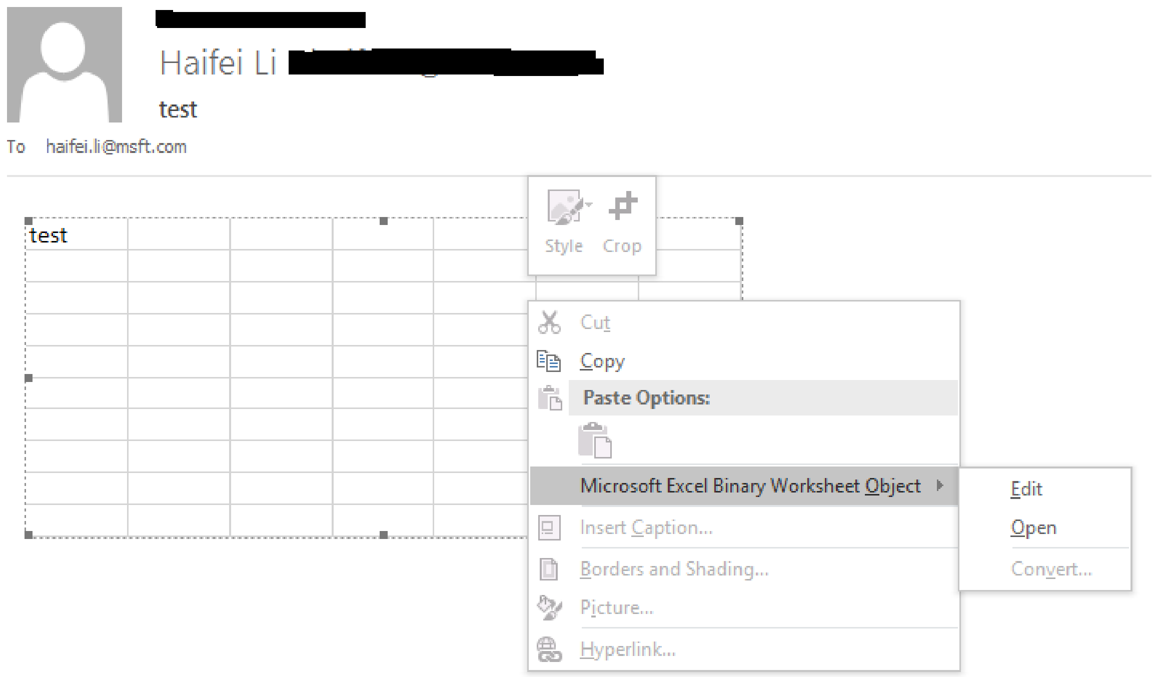
```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F   0123456789ABCDEF
0000h: 78 9F 3E 22 19 17 01 06 90 08 00 04 00 00 00 00  xŸ>"............
0010h: 00 01 00 01 00 01 07 90 06 00 08 00 00 00 E4 04  ..............ä.
0020h: 00 00 00 00 00 00 E8 00 01 08 80 07 00 18 00 00  ......è...€.....
0030h: 00 49 50 4D 2E 4D 69 63 72 6F 73 6F 66 74 20 4D  .IPM.Microsoft M
0040h: 61 69 6C 2E 4E 6F 74 65 00 31 08 01 03 90 06 00  ail.Note.1......
0050h: B8 0F 00 00 24 00 00 00 0B 00 02 00 01 00 00 00  ¸...$...........
0060h: 03 00 26 00 00 00 00 00 0B 00 29 00 00 00 00 00  ..&.......).....
0070h: 1E 00 70 00 01 00 00 00 07 00 00 00 78 78 78 78  ..p.........xxxx
0080h: 78 78 00 00 02 01 71 00 01 00 00 00 16 00 00 00  xx....q.........
```

A malicious winmail.dat, which contains an OLE object, may have the following bytes contained (with the author's comments on the right side).

```
02                                  //level
02 90                               //name, Attachment Rendering Data
06 00                               //type, indicating an OLE object
0E 00 00 00                         //att_length
02 00 00 00 00 00 FF FF FF FF 00 00 00 00     //data
FE 03                               //att_checksum
```

The type "06 00" defines that the "attachment stream" inside the winmail.dat file will be rendered as an OLE object.

Such a feature could allow us to "build" a TNEF email and send it to the user, when the user reads the email, the embedded OLE object will be loaded automatically. Following is an example showing the "Excel Binary Worksheet Object" OLE object is loaded when the user is just reading the email.



P.S.: We may right-click on the object to see the "Microsoft Excel Binary Worksheet Object" menu, indicating the OLE object is indeed loaded.

According to the author's tests, various OLE objects can be loaded via emails; this poses a big security problem. As discussed previously, Outlook has blocked various unsafe attachments, as well as only allowing Office documents to be opened in its Sandbox. However, this feature breaks all the security efforts. I've tested and confirmed that the Flash OLE object (CLSID: *D27CDB6E-AE6D-11cf-96B8-444553540000*) can be loaded via the feature. By packing a Flash exploit in an OLE-enabled TNEF email, an attacker can archive full code execution as long as the victim reads the email.

We use Flash OLE object as an example since Flash (zero-day) exploits are easy to obtain by attackers, but please note that there are other OLE objects may be abused by attacker, as not only Flash but also a number of other OLE objects can be loaded in Outlook.

Another Attack Vector: OLE via MSG

The author has also discovered another way to embed OLE: the .msg file format, though they may share the same code path. With the default configuration, Outlook considers a .msg attachment is safe, thus, it will use the Outlook application itself to open the .msg file even if the user just previews the attachment.

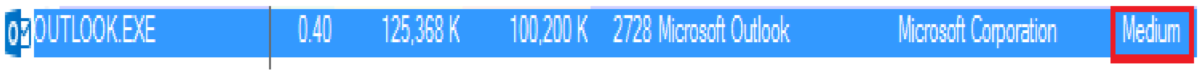
The MSG format is described by Microsoft as well, the sections "*2.2.2.1 Embedded Message Object Storage*", "*2.2.2.2 Custom Attachment Storage*" and "*3.3 Custom Attachment Storage*" describe how to define an OLE object in .msg file, the OLE data should be stored in the sub-storage named as "__substg1.0_3701000D".

The Impact: An Ideal “APT” and Wormable Attacking Technique

As we know, Flash has been proven as an extremely unsafe application during the years; we have seen so many Flash exploits including so many Flash zero-day exploits in the wild. To reduce/mitigate the risks delivered via Flash content, modern browser vendors have been working hard to put Flash content being rendered in a sandboxed environment. For example, on Google Chrome, Flash is run as the Pepper Flash in the Chrome sandbox, for IE11 Flash content is rendered in the Protected Mode which is also an application sandbox, for the newly-released Microsoft Edge browser on Windows 10, all Flash content is rendered in the Enhanced Protected Mode - a much stronger sandbox than the Protected Mode.

Office documents can embed Flash contents as well, which makes Office document seem unsafe to open. However, Microsoft has worked on this – Office documents downloaded from the Internet or delivered via email attachments will be opened in the Office sandbox, this limits the damage caused by malicious Office documents, as we have discussed in previous “Security Mitigations/Enhancements on Outlook” section. In fact, Flash content embedded in Office document will not be rendered at all when in the sandboxed environment.

However, there's no Sandbox for Outlook. Following figure shows Outlook is running with the "Medium" integrity when handling emails - no sandbox at all.



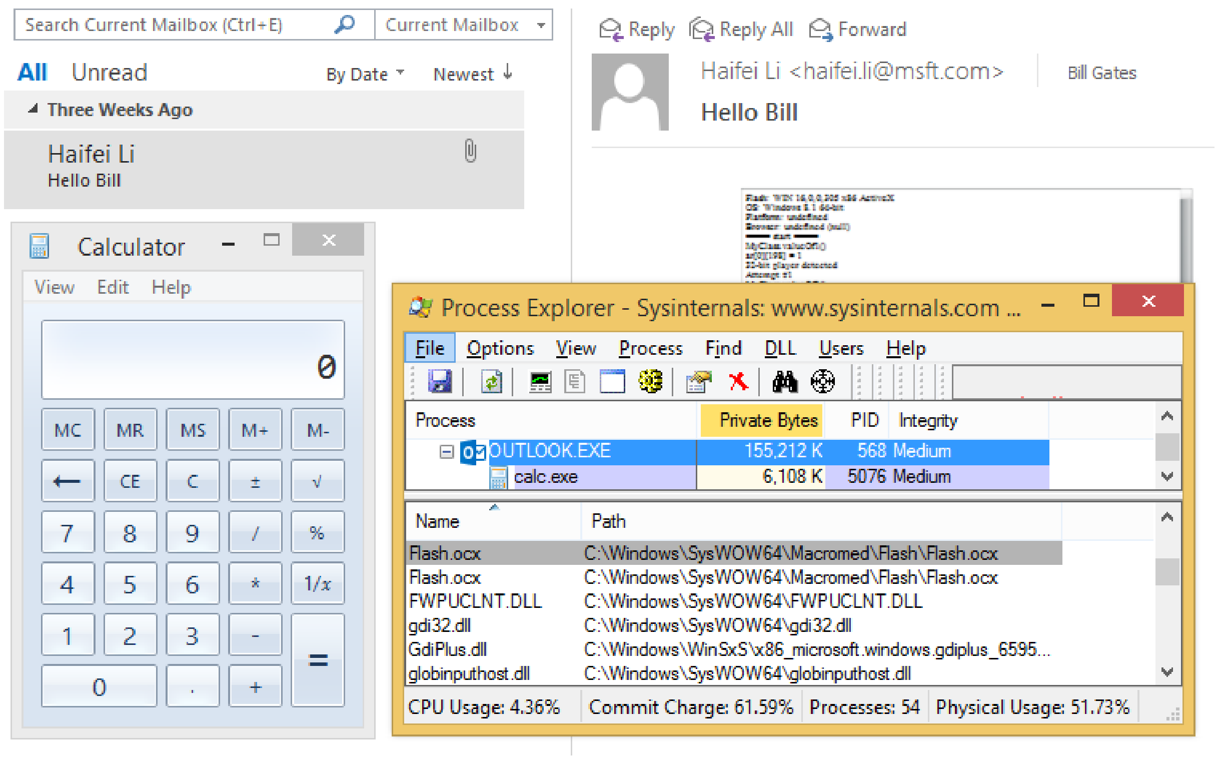
```
OUTLOOK.EXE    0.40    125,368 K    100,200 K    2728 Microsoft Outlook    Microsoft Corporation    Medium
```

What does it mean? It means that if the attacker sends an email to the victim with an embedded Flash exploit (via the “TNEF” format), as long as the victim reads (or we may say, preview) the email, the Flash exploit will be executed in the "outlook.exe" process and it will give the attacker the same privilege of the current user - an ideal way to take control of the victim’s system!

Since Outlook will preview the newest email automatically upon launching, it means that if the attacking email is the newest one, the victim has no choice to avoid being attacked – he/she doesn’t even need to read/preview the attacking email.

Following is the screen captured while the victim just “preview” the email delivered into his/her inbox. It shows:

1. The Windows Calculator was popped up, it means the Flash exploit worked successfully.
2. Outlook process and the calc.exe process are run with “Medium” integrity, means there is no Sandbox on Outlook.
3. The Flash binary (Flash.ocx) is loaded in the Outlook process.



Haifei Li
Hello Bill

Haifei Li <haifei.li@msft.com>
Bill Gates
Hello Bill

```
Flash: WIN 16,0,0,205 x86 ActiveX
OS: Windows 8.1 64-bit
Platform: undefined
Browser: undefined (null)
——— start ———
MyClass.valueOf()
a[0][198] = 1
32-bit player detected
Attempt #1
```

Process Explorer - Sysinternals: www.sysinternals.com ...

Process	Private Bytes	PID	Integrity
OUTLOOK.EXE	155,212 K	568	Medium
calc.exe	6,108 K	5076	Medium

Name	Path
Flash.ocx	C:\Windows\SysWOW64\Macromed\Flash\Flash.ocx
Flash.ocx	C:\Windows\SysWOW64\Macromed\Flash\Flash.ocx
FWPUCLNT.DLL	C:\Windows\SysWOW64\FWPUCLNT.DLL
gdi32.dll	C:\Windows\SysWOW64\gdi32.dll
GdiPlus.dll	C:\Windows\WinSxS\x86_microsoft.windows.gdiplus_6595...
globinputhost.dll	C:\Windows\SysWOW64\globinputhost.dll

CPU Usage: 4.36% | Commit Charge: 61.59% | Processes: 54 | Physical Usage: 51.73%

Even worse, starting from Windows 8, Microsoft has integrated Flash Player (ActiveX version, so can be loaded via OLE) by default, which means that all the Windows 8, Windows 8.1, Windows 10 operating systems are affected by this attack vector by default.

It means that an attacker - who may have a Flash zero-day exploit (considering what we have seen about Flash zero-day attacks in past years, it shouldn't be a rare requirement) - can attack anybody if the victim is using Outlook on a Windows 8/8.1/10 system, or a Windows 7 which has the Flash ActiveX for IE installed.

- **All the attacker needs to know is the email address of the victim**
- **All the victim needs to do is just reading/previewing the email sent from the attacker**

Think about it, an attacker may just need a Flash zero-day exploit (and the email address, of course) to take control of a CEO's computer for a business company - most enterprise users use Outlook every day, then he/she can read all the confidential emails and may do many more. This is absolutely an ideal technology for targeted attacks, especially in an "APT" (advanced persistent threat) era.

Even, an attacker may launch a "worm" based attack by abusing this attack vector – that doesn't usually happen in Windows ecosystem since Vista's release - when compromising one computer via email, the worm may gather all the contacts and then send the same exploit via email to all the contacts to spread itself.

Demonstration

To help readers better analyze the attack vector and understand the impact, the author has made a screen video showing how dangerous it is for this “BadWinmail” attack vector. The video is hosted online at https://youtu.be/ngWVbcLDPm8. In the demo, the author used an old Flash exploit leaked from Hacking Team, the CVE-ID is believed to be CVE-2015-5122. Thus, to ensure the old Flash exploit work, the Flash binary on Windows should less than or equal to 18.0.0.203, as the demo shows.

The attack vector works on all available Windows + Office computing environments, which includes Windows 7/8/8.1/10 having any of the Outlook 2007/2010/2013/2016 installed, prior to the Microsoft’s fix in MS15-131.

Patch & Workarounds

The author has worked with Microsoft to address this serious problem in Outlook since discovered and reported in late October 2015. Microsoft has now addressed the issue on December 8th, 2015, in Microsoft Security Bulletin MS15-131 (CVE-2015-6172). Users are highly recommended to apply the patch immediately.

For users who are not able to apply the official patch for some reason, please follow the Workarounds in MS15-131, where basically it suggests reading emails with plain text only. Additionally, the author would suggest setting an “Office Kill-bit” register key to prevent Outlook from loading the “highly-risky” Flash content, the blocked CLSID is *D27CDB6E-AE6D-11cf-96B8-444553540000*. The author has confirmed that by setting the following registry key, Outlook will not load Flash content anymore.

```
Windows Registry Editor Version 5.00
[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Office\Common\COM Compatibility\{D27CDB6E-AE6D-11cf-96B8-444553540000}]
"Compatibility Flags"=dword:00000400
```

Conclusion

In this report, the author disclosed a novel attack vector to attack Outlook users via emails, which the author named as **BadWinmail**. Specifically, we disclosed that a Flash (or other types of) exploit can be packed and delivered via a TNEF email (or MSG attachment). The most serious impact is that the exploit will get executed as long as the Outlook user reads/previews the attacking email. Because there is no sandbox on Outlook, it allows the attacker to take control of the victim’s computer immediately.

BadWinmail is an ideal attacking technique for targeted/APT attacks because of its severity and the nature of email-based attacks - all the attacker needs to know is the victim’s email address. It’s a “killer” exploit-delivering method as usual tricks such as delivering via email attachments or delivering via URLs (in email bodies) require additional user interactions and are protected by various application sandboxes. It’s also a wormable issue rarely seen on Windows platform nowadays.