Hunting for bugs, catching dragons

Nicolas Joly - @n_joly
MSRC Vulnerabilities and Mitigations Team
Attacking Outlook with 💌s
Outlook exploits?

• Uncommon, not seen in the wild for a while
  • I Love You / Love Letter, early 2000

• Badwinmail reported by Haifei Li in late 2015:
  • Showed how to load Flash in Outlook
  • Leveraged a known Flash vulnerability to prove RCE

• Ryan Hanson’s amazing research submitted in late 2016:
  • Issues with the RTF format
  • COMs and Monikers
  • Some cool Outlook features

• Abusing Word features
  • Embedding an EPS font (CVE-2015-2545)?
### Quick summary of the attack surface

<table>
<thead>
<tr>
<th>Email protocols</th>
<th>Email parsing</th>
<th>Spoofing</th>
</tr>
</thead>
<tbody>
<tr>
<td>SMTP</td>
<td>MIME parsing</td>
<td>Certs issues</td>
</tr>
<tr>
<td>POP3/IMAP</td>
<td>HTML / RTF</td>
<td>Name spoofing</td>
</tr>
<tr>
<td>Exchange Active Sync</td>
<td>Pictures (GDI or Office stacks)</td>
<td>S/MIME</td>
</tr>
<tr>
<td>Exchange Web Services</td>
<td>Fonts</td>
<td>DRMs</td>
</tr>
<tr>
<td>Autodiscover</td>
<td>OLE Objects</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Calendars, iCals, vCards, contacts</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Attachments</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>TNEF, MAPI properties</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td></td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Misc</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>Macros</td>
<td>...</td>
</tr>
<tr>
<td></td>
<td>SensePost’s Ruler (rules, scripts, homepage)</td>
<td>...</td>
</tr>
</tbody>
</table>
**What this talk covers**

<table>
<thead>
<tr>
<th>Email parsing</th>
<th>Email protocols</th>
<th>Spoofing</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIME-parsing</td>
<td>SMTP</td>
<td>Certs-issues</td>
</tr>
<tr>
<td>HTML / RTF</td>
<td>POP3/IMAP</td>
<td>Name-spoofing</td>
</tr>
<tr>
<td>Pictures (GDI or Office stacks)</td>
<td>Exchange-Active-Sync</td>
<td>S/MIME</td>
</tr>
<tr>
<td>Fonts</td>
<td>Exchange-Web-Services</td>
<td>DRM's</td>
</tr>
<tr>
<td></td>
<td>Autodiscover</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td><strong>OLE Objects</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Calendars, iCals, vCards, contacts</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Attachments</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TNEF, MAPI properties</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Misc</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Macros</td>
<td>SensePost’s Ruler (rules, scripts, homepage)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>
Why this talk?

• Exploits for Outlook exist but we only occasionally receive reports of dragons outstanding issues
• Why aren’t researchers reporting to us?
  • Lack of public research, blog posts describing issues?
  • Lack of interest in the area?
  • Symbols unavailable for Office?
• How can we help our finders?
• Let’s talk about our own research!
• Note:
  • The vulnerabilities discussed in the following slides have all been resolved
How email message formats affect Internet email messages in Outlook

The use of TNEF is commonly affected by settings in Outlook that are referred to as Microsoft Outlook Rich Text Format (RTF). Rich Text Format and TNEF are not exactly the same, but they are closely related.

A TNEF-encoded message contains a plain text version of the message, and a binary attachment that "packages" various other parts of the original message. In binary attachment is named Winmail.dat, and may include:

- The formatted text version of the message
- OLE objects (for example, embedded Excel and Word Office documents).
- Special Outlook features (for example, custom forms, voting buttons, and meeting requests).
- Regular file attachments that were added to the original message.

https://support.microsoft.com/sq-al/help/290809
What's a TNEF email?

What's this?
TNEF specifications

- **[MS-OXTNEF]**
- Sequence of objects, containers and properties
- Easy to parse
- Developed an 010 template
- Might release in the future

<table>
<thead>
<tr>
<th>Name</th>
<th>Value</th>
<th>Start</th>
<th>Size</th>
<th>Color</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>struct FILE file</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>struct ATTRIBUTE attr[0]</td>
<td>0h</td>
<td>6h</td>
<td>6h</td>
<td>Fg</td>
<td>ATTNENVERSION</td>
</tr>
<tr>
<td>struct ATTRIBUTE attr[1]</td>
<td>6h</td>
<td>12h</td>
<td>8h</td>
<td>Fg</td>
<td>ATTOEMCODEPAGE</td>
</tr>
<tr>
<td>struct ATTRIBUTE attr[2]</td>
<td>12h</td>
<td>20h</td>
<td>8h</td>
<td>Fg</td>
<td>ATMESSAGECLASS</td>
</tr>
<tr>
<td>struct ATTRIBUTE attr[3]</td>
<td>20h</td>
<td>28h</td>
<td>8h</td>
<td>Fg</td>
<td>ATTMAPROPS</td>
</tr>
<tr>
<td>char level</td>
<td>1</td>
<td>1h</td>
<td></td>
<td>Fg</td>
<td></td>
</tr>
<tr>
<td>ushort name</td>
<td>36867</td>
<td>54h</td>
<td>2h</td>
<td>Fg</td>
<td></td>
</tr>
<tr>
<td>ushort type</td>
<td>6</td>
<td>56h</td>
<td>2h</td>
<td>Fg</td>
<td></td>
</tr>
<tr>
<td>int length</td>
<td>1720h</td>
<td>58h</td>
<td>4h</td>
<td>Fg</td>
<td></td>
</tr>
<tr>
<td>int nProps</td>
<td>70</td>
<td>5Ch</td>
<td>4h</td>
<td>Fg</td>
<td></td>
</tr>
<tr>
<td>struct TNEFProperty prop[0]</td>
<td></td>
<td>60h</td>
<td>8h</td>
<td>Fg</td>
<td>PidTagAlternateRecipientAllowed BOOL</td>
</tr>
<tr>
<td>struct TNEFProperty prop[1]</td>
<td></td>
<td>68h</td>
<td>8h</td>
<td>Fg</td>
<td>PidTagPriority INT</td>
</tr>
<tr>
<td>struct TNEFProperty prop[2]</td>
<td></td>
<td>70h</td>
<td>8h</td>
<td>Fg</td>
<td>PidTagReadReceiptRequested BOOL</td>
</tr>
<tr>
<td>struct TNEFProperty prop[3]</td>
<td></td>
<td>78h</td>
<td>10h</td>
<td>Fg</td>
<td>PidTagConversationTopic String8</td>
</tr>
<tr>
<td>struct TNEFProperty prop[4]</td>
<td></td>
<td>88h</td>
<td>24h</td>
<td>Fg</td>
<td>PidTagConversationIndex BINARY</td>
</tr>
<tr>
<td>struct TNEFProperty prop[5]</td>
<td></td>
<td>ACh</td>
<td>8h</td>
<td>Fg</td>
<td>PidTagDeleteAfterSubmit BOOL</td>
</tr>
</tbody>
</table>
TNEF or RTF - Rich Text Format?

- One special property
  - PR_RTF_COMPRESSED = 0x10090102
- Various encodings
  - LZfu - Compressed RTF, default
  - MELA - Plain RTF, easy to change
- Bit flipping this field will generally result in a broken email
- Use instead the Outlook Interop Library and send TNEF emails programmatically
Examples of issues affecting Outlook

- **CVE-2017-0106**: Introduced by the `\template` keyword:
  - Remotely (and locally) loads files (http, smb)
  - Also loads embedded objects, like Flash
  - Dramatically extends the attack surface by allowing all the Word supported formats (docx, doc...)
- **CVE-2018-0794**
  - Cyclic reference in the template names leading to Use After Free
    - a.rtf => b.rtf => \not found\a.rtf
- **Resolution?**
  - `\template` no longer supported in Outlook
Examples of issues affecting Outlook

- **CVE-2017-0106**
  - Introduced by the template keyword:
  - Remotely (and locally) loads files (http, smb)
  - Also loads embedded objects, like Flash
  - Dramatically extends the attack surface by allowing all the Word supported formats (docx, doc…)

- **CVE-2018-0794**
  - Cyclic reference in the template names leading to Use After Free
  - a.rtf => b.rtf => a.rtf

**Resolution?**
- Template no longer supported in Outlook
Outlook Interop, how to build an email

• Provides an API to build your own emails
  • Example, create a MailItem, change the MessageClass to IPM.Contact and send it
  • See the contact form appear in the preview pane

• Each class has its own properties/features

<table>
<thead>
<tr>
<th>Item</th>
<th>Default folder</th>
<th>Default message class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contact</td>
<td>Contacts</td>
<td>IPM.Contact</td>
</tr>
<tr>
<td>Task</td>
<td>Tasks</td>
<td>IPM.Task</td>
</tr>
<tr>
<td>Appointment</td>
<td>Calendar</td>
<td>IPM.Appointment</td>
</tr>
<tr>
<td>Note</td>
<td>Notes</td>
<td>IPM.StickyNote</td>
</tr>
<tr>
<td>Journal Entry</td>
<td>Journal</td>
<td>IPM.Activity</td>
</tr>
<tr>
<td>Mail</td>
<td>Inbox</td>
<td>IPM.Note</td>
</tr>
</tbody>
</table>
Outlook Interop, how to build an email

- Provides an API to build your own emails
- Example, create a `MailItem`, change the `MessageClass` to `IPM.Contact` and send it
- See the contact form appear in the preview pane
- Each class has its own properties/features

```csharp
static void Main(string[] args)
{
    var objOutlook = new Application();
    var accounts = objOutlook.Session.Accounts;
    MailItem mic = (MailItem)(objOutlook.CreateItem(OLItemType.olMailItem));
    mic.MessageClass = "IPM.Contact";
}```
Other class names and bugs related

• Some examples listed here https://docs.microsoft.com/en-us/office/vba/outlook/concepts/forms/item-types-and-message-classes

• Create a mail item and change its class to IPM.Remote
  • Used to trigger a null pointer in the preview pane
  • Reported by Etienne Stalmans and fixed as vNext

• IPM.Document.*, aka “freedocs”
  • CVE-2017-0204, Office documents open without Protected Mode, Ryan Hanson
  • CVE-2017-8571, Office documents open without user interaction

```csharp
static void Main(string[] args)
{
    MailItem mic = (MailItem)(objOutlook.CreateItem(OlItemType.olMailItem));
mic.RTFBody = System.IO.File.ReadAllBytes(@"C:\temp\empty.rtf");
mic.Subject = "Important Email";
mic.Importance = OlImportance.olImportanceHigh;
mic.Attachments.Add(@"C:\temp\hello.docx", OlAttachmentType.olOLE);
mic.Send();
}
```
Other class names and bugs related
• Use Interop to create Outlook .msg files

```csharp
static void Main(string[] args)
{
    var objOutlook = new Application();
    var accounts = objOutlook.Session.Accounts;
    MailItem mic = (MailItem)objOutlook.CreateItem(OlItemType.olMailItem);
    mic.MessageClass = "IPM.Contact";

    mic.SaveAs("email.msg");
}```
Use Interop to create Outlook .msg files in TNEF / .MSG format
MAPI properties, TNEF and .MSG?

• TNEF is “a hierarchy of rich message properties”, a succession of particular attributes, called MAPI properties, forming a stream.
  • Example below with a Task message:

<table>
<thead>
<tr>
<th>Name</th>
<th>Other Names</th>
<th>Tag</th>
<th>Type</th>
<th>Value</th>
<th>Value (alternate view)</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR_ICON_INDEX</td>
<td>PidTagIconIndex, ptagIconIndex</td>
<td>0x10080003</td>
<td>PT_LONG</td>
<td>1282</td>
<td>0x502</td>
</tr>
<tr>
<td>PR_IMPORTANCE</td>
<td>PidTagImportance, ptagImportance</td>
<td>0x00170003</td>
<td>PT_LONG</td>
<td>1</td>
<td>0x1</td>
</tr>
<tr>
<td>PR&lt;Internet CpID&gt;</td>
<td>PidTagInternetCodepage, ptagInt...</td>
<td>0x3FDE0003</td>
<td>PT_LONG</td>
<td>28591</td>
<td>0x6FAF</td>
</tr>
<tr>
<td>PR_LAST_MODIFICATION_TIME</td>
<td>PidTagLastModificationTime, ptag...</td>
<td>0x30080040</td>
<td>PT_SYSTIME</td>
<td>10:36:08.221 AM 26/03/2019</td>
<td></td>
</tr>
<tr>
<td>PR_LAST_MODIFIER_NAME_W</td>
<td>PidTagLastModiﬁerName, PR_LAS...</td>
<td>0x3FFA001F</td>
<td>PT_UNICODE</td>
<td><a href="mailto:Nicolas.Joly@microsoft.com">Nicolas.Joly@microsoft.com</a></td>
<td></td>
</tr>
<tr>
<td>PR_MAPPING_SIGNATURE</td>
<td>PidTagMappingSignature, ptagMap...</td>
<td>0x0FF80102</td>
<td>PT_BINARY</td>
<td>cb: 16 lpb: 973A0F6F6454AC44B3E...</td>
<td></td>
</tr>
<tr>
<td>PRMDB_PROVIDER</td>
<td>PidTagStoreProvider</td>
<td>0x34140102</td>
<td>PT_BINARY</td>
<td>cb: 16 lpb: 5494A1C0297F1018A58...</td>
<td></td>
</tr>
<tr>
<td>PR_MESSAGE_ATTACHMENTS</td>
<td>PidTagMessageAttachments, ptagM...</td>
<td>0x0E12000D</td>
<td>PT_OBJECT</td>
<td>Object</td>
<td></td>
</tr>
<tr>
<td>PR_MESSAGE_CLASS_W</td>
<td>PidTagMessageClass, PR_MESSAG...</td>
<td>0x001A001F</td>
<td>PT_UNICODE</td>
<td>IPM.Task</td>
<td></td>
</tr>
<tr>
<td>PR MESSAGE DELIVERY_TIME</td>
<td>PidTagMessageDeliveryTime</td>
<td>0x0E060040</td>
<td>PT_SYSTIME</td>
<td>03:54:05.120 PM 22/09/2017</td>
<td></td>
</tr>
<tr>
<td>PR MESSAGE FLAGS</td>
<td>PidTagMessageFlags, ptagMessage...</td>
<td>0x0E070003</td>
<td>PT_LONG</td>
<td>17</td>
<td>0x11</td>
</tr>
</tbody>
</table>

• A MAPI property is defined by a PID, a Type and a value
A .MSG is an OLE Storage document, with streams matching MAPI properties, and sub-storages matching MAPI objects.

Properties are defined in the __properties_version1.0 stream:

Small properties (integers, bools...) are defined in that stream.

Arrays, strings or other objects are defined in their own streams. For example the message class property is defined in __substg1.0_001A001F:
From Haifei’s research, we know that we can embed objects.

How are these processed exactly? Can we load scripts?

The OLE storages are easy to manipulate.
  - What else is hiding in there?

Test case, insert an object in an email, put some breakpoints on the usual COM interoperability functions in ole32.dll.
  - ReadClassStg, OpenStorageEx, etc.
  - Pictures and links are processed differently.
  - With an object link (CLSID_StdOleLink) we can hit OleLoad().
• From Haifei's research, we know that we can embed objects in an email.
• How are these processed exactly? Can we load scripts?
• The OLE storages are easy to manipulate.
• What else is hiding in there?
• Test case, insert an object in an email, put some breakpoints on the usual COM interoperability functions in ole32.dll.
• ReadClassStg, OpenStorageEx, etc.
• Pictures and links are processed differently.
• With an object link (CLSID_StdOleLink) we can hit OleLoad().
Invoking COM Monikers from a .MSG

- Create an RTF email and insert a link (you may want to use an older version of Outlook)
- Save it to a .MSG storage
- Locate the OLE storage specifically created
- Create a new \x01Ole stream underneath
- Hit OleLoad()
- Instantiate monikers
- Profit!
Unmarshalling COM Monikers

1615   // read size LONG followed by persistent moniker
1616   STDAPI ReadMonikerStm (LPSTREAM pstm, LPMONIKER* ppmk)
1617   {  

This function can be used to load an object that supports the IPersistStream interface.

1620   if ((error = StRead (pstm, &cb, sizeof(DWORD))) != NOERROR)
1621       return error;
1622   
1623   if (cb == NULL)
1624       return NOERROR;
1625   
1626   return OleLoadFromStream (pstm, IID_IMoniker, (LPLPVOID) ppmk);  
1627 }
Example: FileMoniker, CVE-2018-0950

Combine with an SMB vulnerability, and you've got some real fun.
More with the FileMoniker

- Ever looked at CFileMoniker::RestoreShellLink in Ole32?

  ```c
  NT_VERIFY(S_OK == m_pShellLink->QueryInterface(IID_IPersistStream,
                      (void**)&pps));

  memset(&ll0, 0, sizeof(ll0));
  NT_VERIFY(S_OK == pstm->seek(ll0, STREAM_SEEK_SET, &ull));
  NT_ASSERT(u11.LowPart == 0 && u11.HighPart == 0);

  if (S_OK != hr) pps->Load(pstm));
  ```

- FileMonikers support .LNK shortcuts:
  - CVE-2018-0825, integer overflow in StructuredQuery
  - Load dlls with CVE-2017-8464?
Another example: OBJREF

- The objref moniker allows unmarshalling arbitrary objects on the IUnknown interface:

```c
STDMETHODIMP CObjrefMoniker::Load(IStream *pStream)
{
    HRESULT hr;
    ULONG cbRead;

    mnkDebugOut((DEB_ITRACE, "%s\n", this, pStream));

    if (!pStream)
        return E_INVALIDARG;

    // Unmarshal the object we're wrapping
    return CoUnmarshalInterface(pStream, IID_IUnknown,(LPVOID *) &m_pUnk);
}
```
OBJREF – Building the exploit chain

• Still far from calc:

CoUnmarshalInterface

IPersistStreamInit::Load

Flash?

XML Feed Moniker

Load a VB Script

VB 0day

Calc with a…

MSVIDCTL – BDATuner.Component Types

IPersistStreamInit::Load

Initialize with HTML data
Multiple bugs fixed in this attack

- Restrict the objects loaded by msvidctl.dll
  - CVE-2016-0142, CVE-2016-7248, CVE-2018-0881

- Prevent the objref and XML Feed Moniker objects from loading in Office via the COM Activation filter

- Do not load OLE objects in the pane
  - CVE-2018-0950

- Fix another VBScript bug
  - And block VBScript as well in the Activation Filter (recent Office branches only)

- Restrict objects loaded by DiagnosticsHub.StandardCollector service
  - CVE-2018-0824
  - How is this related at all?
From the preview pane to system

• This issue does not only apply to Office, COM marshalling is extensively used by the system

• The attack surface is quite large:
  • VARIANTs
  • SAFEARRAYs

• We just need to find a system COM with a method that accepts such argument
SafeArrays are generic

- They can contain bytes, integers, strings, all sorts of objects:
  - Including VT_UNKNOWN objects
- Look at the logic in LPSAFEARRAY_Unmarshal
  - We can quickly reach another CoUnmarshalInterface
  - And replay the attack
Ex: Diagnostics Hub Standard Collector Service

```csharp
namespace Microsoft
{
    namespace DiagnosticsHub
    {
        namespace StandardCollector
        {
            /// <summary>
            /// ETW-specific implementation of <see cref="ETWCollectorService"/>.
            /// </summary>
            class EtwCollectionSession :
            {
                public ICollectionSession,
                public ICollectionSessionEx,
                public IDebuggerCollectionSession,
                public ISupportEventArgs,
                public CModuleRefCount
                {
                    ...
                }
            }

            HRESULT STDMETHODCALLTYPE GetGraphDataUpdates(
                /* [in] */ __RPC__in REFGUID agentId,
                /* [in] */ __RPC__in SAFEARRAY * counterIdAsBstrs,
                /* [retval][out] */ __RPC__out struct GraphDataUpdates *result);

            • Just calling GetGraphDataUpdates is enough to trigger the chain
```
Attacking Exchange with 💌s
Attacking Exchange with emails

• Where to start?
  • ShadowBrokers’ EnglishmansDentist targeting Exchange 2003
  • Voicemail Transcription RCE via .NET deserialization (CVE-2018-8302, not an email scenario)

• Various attack scenarios:
  • Are we already authenticated?
  • Are we playing with memory corruptions? Replaying tokens? Web issues?

• Is everything handled by managed code?
  • Looking at Exchange Onprem gives a good idea of what’s running
  • exRPC32.dll, what’s that?

• Some tools:
  • MFCMapi
MFCMapi, your best friend
exRPC and MAPI properties

• Loaded by MSExchangeDelivery.exe

• Several functions parse properties found in TNEF emails:
  • A property has a pid and a type:
    • PT_STRING8
    • PT_INT
    • PT_BINARY...

• Is there any bug left?

```c
switch (pid) {
    default:
        if (piprv)
            piprv[iprvRpc] = iprv;
        pprvRpc[iprvRpc++] = pprv[iprv];
        break;
    case PROP_ID(ptagSubject):
        if (PROP_TYPE(ptag) == PT_STRING8)
            
        else if (PROP_TYPE(ptag) == PT_UNICODE)
            
        else
            
            ecProblem = ecComputed;
            goto Problem;
        
        break;
```
EcParseEntryId

- Called for pid ptagConflictEntryId, only accepts a byte array
- No checks on the property type
- Supported variants are stored on 0x18 bytes on 64-bit
  - Scalars at offset +8
  - Pointers at +0x10
- Probably a DoS at worst

```c
case PROP_ID(ptagConflictEntryId):
    BOOL fLongTerm = fFalse;
    BOOL fMessage = fFalse;
    BOOL fWacky = fFalse;
    LTID *pltidFolder = NULL;
    LTID *pltidMessage = NULL;
    GID gidFolder = { 0 };
    GID gidMessage = { 0 };

    // Do a quick parse to verify EntryId is what we expect
    ec = LOGONOBJ::EcParseEntryId(
        prvr[irvr].Value.bin.cb,
        (ENTRYID *)prvr[irvr].Value.bin.lp,
        &fLongTerm,
        &fMessage,
        &fWacky,
        NULL,
        NULL,
        NULL);
    if (ec)
    {
        ecProblem = ec;
        goto Problem;
    }
```
EcMakeFidlFromFEL

- Called for pid PostReplyFolderEntries, only accepts a byte array
- A loop where data is read and partially written in the original buffer
- No bound check

- We can alter one bit OOB in the last entry
  - And [rdi+18h], ax with ax = 7FFFh
  - So what do you think I did?
    - Nothing, was too hard 😊
Outcome of that research

• Are these issues exploitable? Very unlikely, but proves the concept
• Other issues found, essentially type confusions
  • These are likely exploitable after authentication, DoS otherwise
• Found also some issues affecting the .NET binaries
  • Null pointers leading to temporary DoS
• Uninitialized memory while parsing rules
  • PR_EXTENDED_RULE_ACTIONS
  • Likely exploitable but would need an infoleak first
• Other interesting components to look at, think OWA too

What else can YOU find?
References

- https://www.fireeye.com/blog/threat-research/2015/09/attack_exploitingmi.html
- https://docs.microsoft.com/en-us/openspecs/windows_protocols/ms-oaut/3fe7db9f-5803-4dc4-9d14-5d4253f5461f
- https://docs.microsoft.com/en-us/openspecs/exchange_server_protocols/ms-oxtnef/1f0544d7-30b7-4194-b58fadcd82f3e53bb
- https://github.com/stepheneggriffin/mfcma
- CVE-2017-8506, demo on slide3
Thanks all!

@n_joly