

XLL PHISHING

Utilizing Microsoft Excel Add-In's for Initial Access

WHOAMI

Alex Reid
OSCP, OSEP, RTJC
Red Teamer and Tool Developer

~2.5 years in cyber / red teaming / development

2x contributor to CobaltStrike Community Kit

Interested in Windows malware development and exploitation

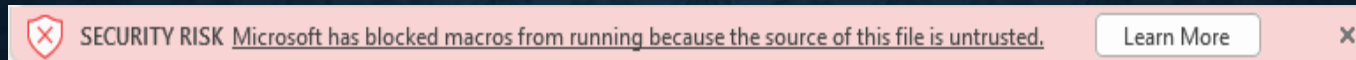
Primarily develop in C, work in Python3 / Powershell when the opportunity arises

GitHub: <https://github.com/Octoberfest7>



“NECESSITY IS THE MOTHER OF INVENTION”

- VBA Macros embedded within Office docs are well known and ubiquitous with phishing
- Microsoft is FINALLY taking steps to address this attack vector
- Office files containing macros obtained via email or internet are now hard-disabled:
- “With this change, when users open a file that came from the internet, such as an email attachment, and that file contains macros, the following message will be displayed:”¹



1. <https://learn.microsoft.com/en-us/deployoffice/security/internet-macros-blocked>

THEORY OF PHISHING FOR ACCESS

- What makes a good phishing for access vector? Considerations:
 - Complexity- How many steps are required by the victim?
 - Specificity- Is the vector architecture specific? Dependent on third-party software?
 - Delivery- Can it be attached to an email? Downloaded from a webserver?
 - Attack Surface Reduction- Application whitelisting? Network protections? Other ASR rules?
 - Detection- AV/EDR on endpoints?
- Factors can compound; an adaptation to overcome one problem may cause another

SETTING THE STAGE

- This research was performed with a fictional target organization in mind that employs several industry standard defensive measures:
 - Email filtering rules preventing dangerous file types from being received
 - A network proxy that blacklists certain file types from being downloaded
 - Application whitelisting on endpoints that limit potential payload formats
 - Data Loss Prevention (DLP) measures prevent mounting of external storage devices
 - Microsoft Defender for Endpoint (MDE) deployed and active

WHAT IS AN XLL?



- An XLL is a DLL with a special entry point that Microsoft Excel recognizes and can load
- Been around since Microsoft Excel 97!
- Create new user-defined functions and features in Excel
- XLL's can be written in C or C++ (and maybe C# with some extra work)

```
xlAddInManagerInfo/xlAddInManagerInfo12  
xlAutoAdd  
xlAutoClose  
xlAutoFree/xlAutoFree12  
xlAutoOpen  
xlAutoRegister/xlAutoRegister12  
xlAutoRemove
```


XLL'S AND THREAT ACTORS

- A quick Google search shows research back in 2017 on using XLL's and WLL's for persistence
- Lots of articles on threat actors and APT's utilizing XLL's from 2020 -> present
- Agent Tesla, Dridex, RedLine, Hancitor, BazaLoader, Raccoon Stealer²³⁴⁵

2. <https://unit42.paloaltonetworks.com/excel-add-ins-malicious-xll-files-agent-tesla/>

3. <https://threatresearch.ext.hp.com/how-attackers-use-xll-malware-to-infect-systems/>

4. <https://isc.sans.edu/forums/diary/Hancitor+tries+XLL+as+initial+malware+file/27618/>

5. <https://www.zdnet.com/article/theres-been-a-big-rise-in-phishing-attacks-using-microsoft-excel-xll-add-ins/>

AN INITIAL INVENTORY

Pros

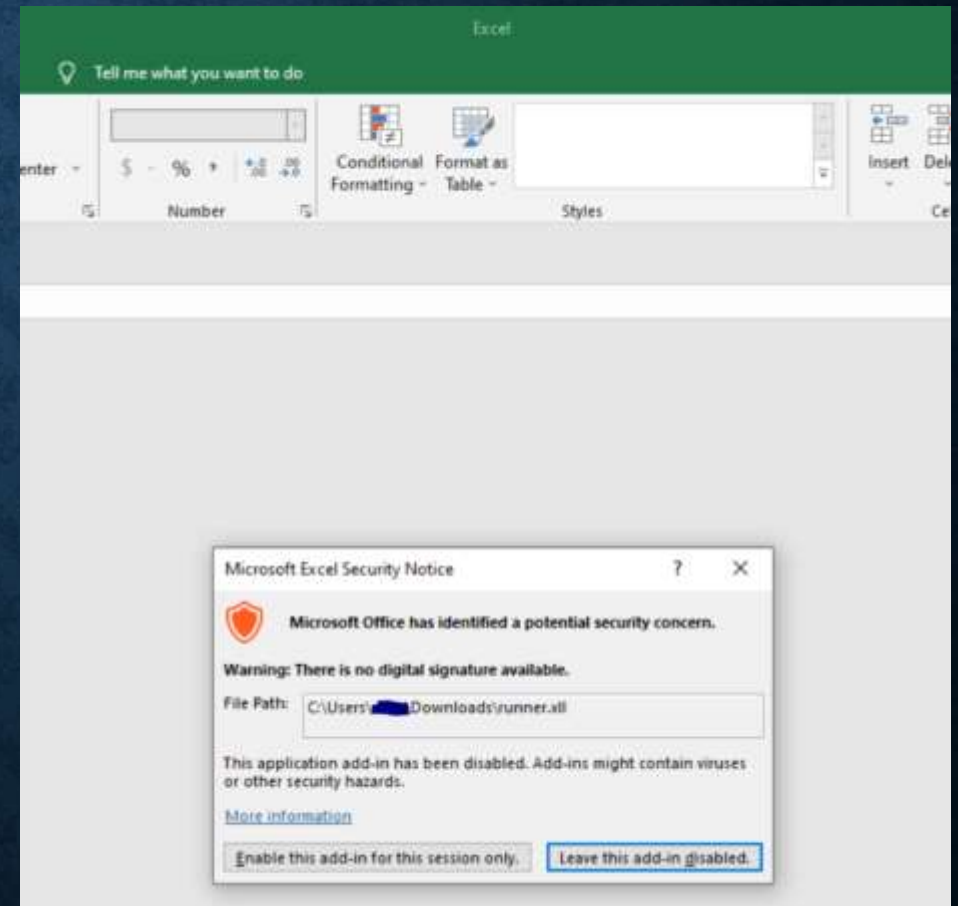
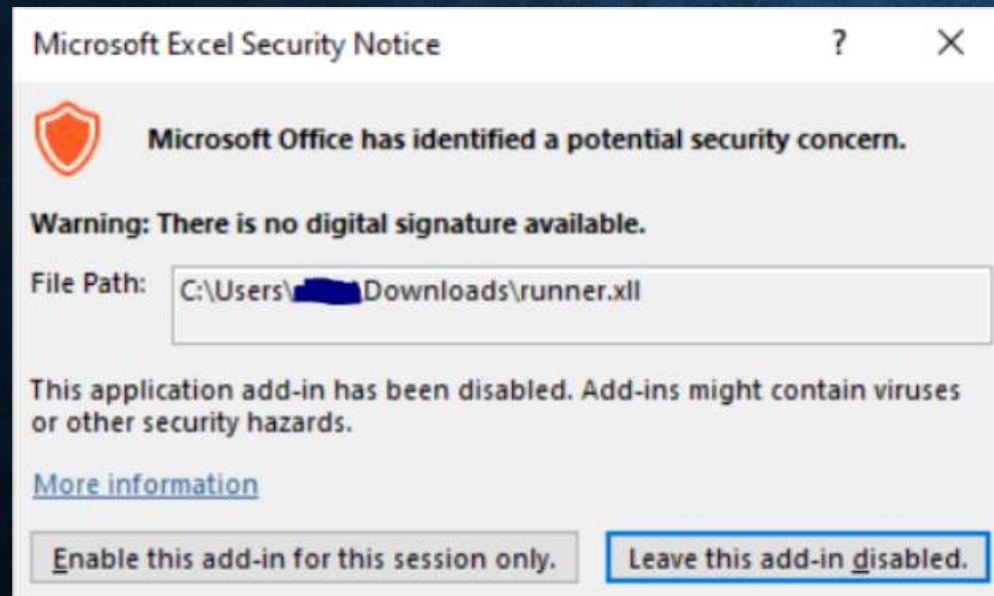
- XLL's are ran by Excel; a trusted application
- Excel is commonly encountered
- **Written in C or C++; can utilize much more robust malware TTP's than in VBA**

Cons

- XLL's aren't commonly encountered
- They are executables; MZ header
 - This can limit delivery options
- Architecture specific; x86 XLL will NOT run in x64 Excel!

HOW TO EXECUTE?

- File explorer / Desktop: Just double click!
- One dialogue stands between user and code execution



DELIVERY METHODS

- Email Attachment:
 - Organizations and email clients have very robust protections nowadays
 - Example rules:
 - Block executable attachments (EXE, DLL, XLL, etc)
 - Block certain container files (ISO, IMG, etc)
 - Examine ZIP files and block those containing executables
 - Block password protected ZIP's
- No good...

DELIVERY METHODS

- Web Delivery
 - Less ideal; email target with a link to your webserver to pull down hosted XLL
 - Domain reputation? Network proxy?
 - Blocked file types? Executables (EXE, DLL, XLL, etc) still aren't allowed...
 - What about containers? Rules may be different than those applied to emails
 - (Notional) Success!
 - ISO/IMG are still no good due to DLP; ZIP's are allowed!
 - Wait... Executables are blocked, but executables in ZIP's are allowed?
 - Defensive policies can be peculiar; poke and prod and you might just find a way

DELIVERY METHODS

- A sneaky side-bar...
- Network inspection of traffic leaving an organization can raise eyebrows
 - Someone is downloading a zip (from an obscure domain)? What is it?
- Utilize Apache mod_rewrite to modify traffic on the redirector
 - Phishing link in email: <https://mydomain.com/jobs/relay/ITsalary>
 - GET request modified on redirector -> <https://mydomain.com/jobs/relay/ITsalary.zip>
 - Doesn't look like a request for a ZIP leaving victim network, but they receive a ZIP

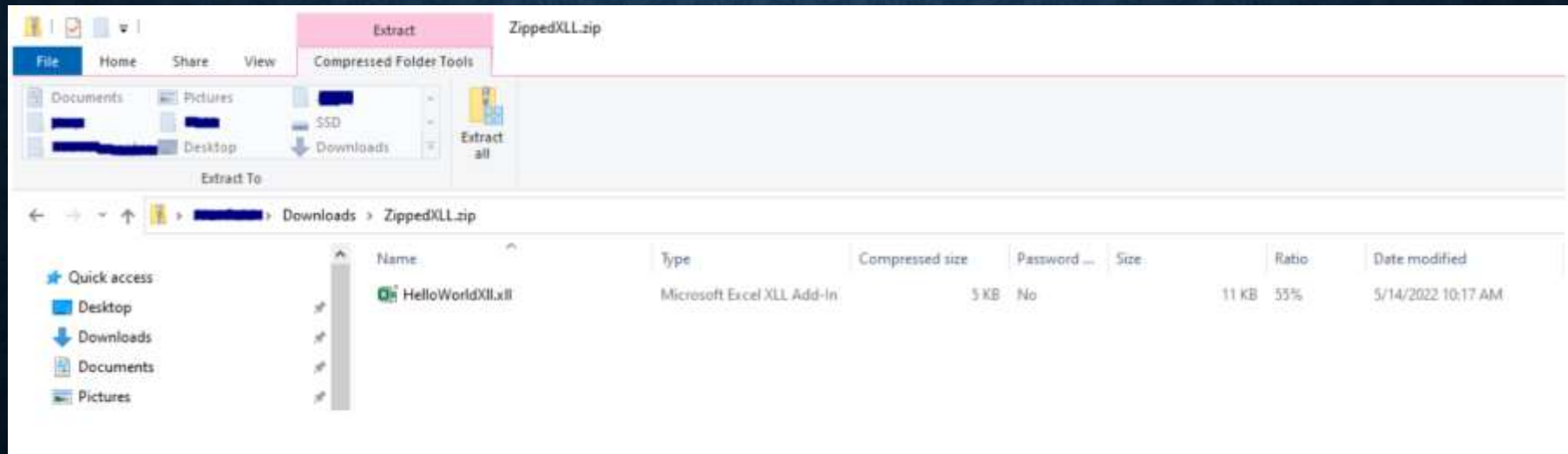
```
RewriteCond %{REQUEST_URI} "^(\/[a-zA-Z0-9_\-]{2,})$"
RewriteRule \([a-zA-Z0-9_\-]+\)$ https://[REDACTED]/$1.zip [P]
ProxyPassReverse / https://[REDACTED]
```

```
"GET / [REDACTED]/ITsalary.zip HTTP/1.1" 200 444030
```

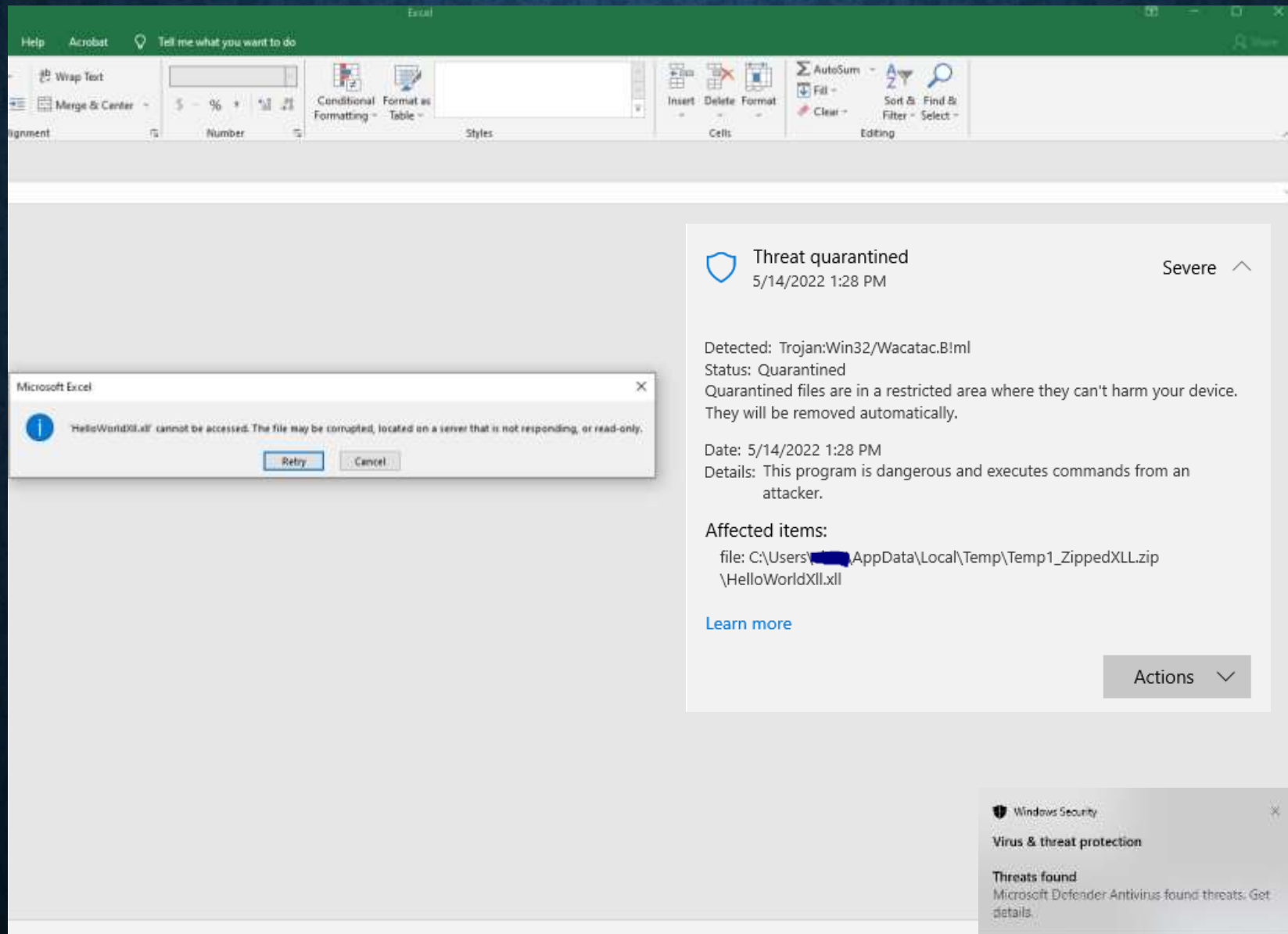
```
"GET /jobs/relay/ITsalary HTTP/1.1" 200 444030
```


ZIP FILES AND EXECUTION

- ZIP's are a strong choice because:
 - Natively compatible with Windows
 - Can be downloaded from the internet by our organization
 - Add little complexity to the attack
 - Victim's can click the ZIP file in the browser download bar and file explorer will open it up:



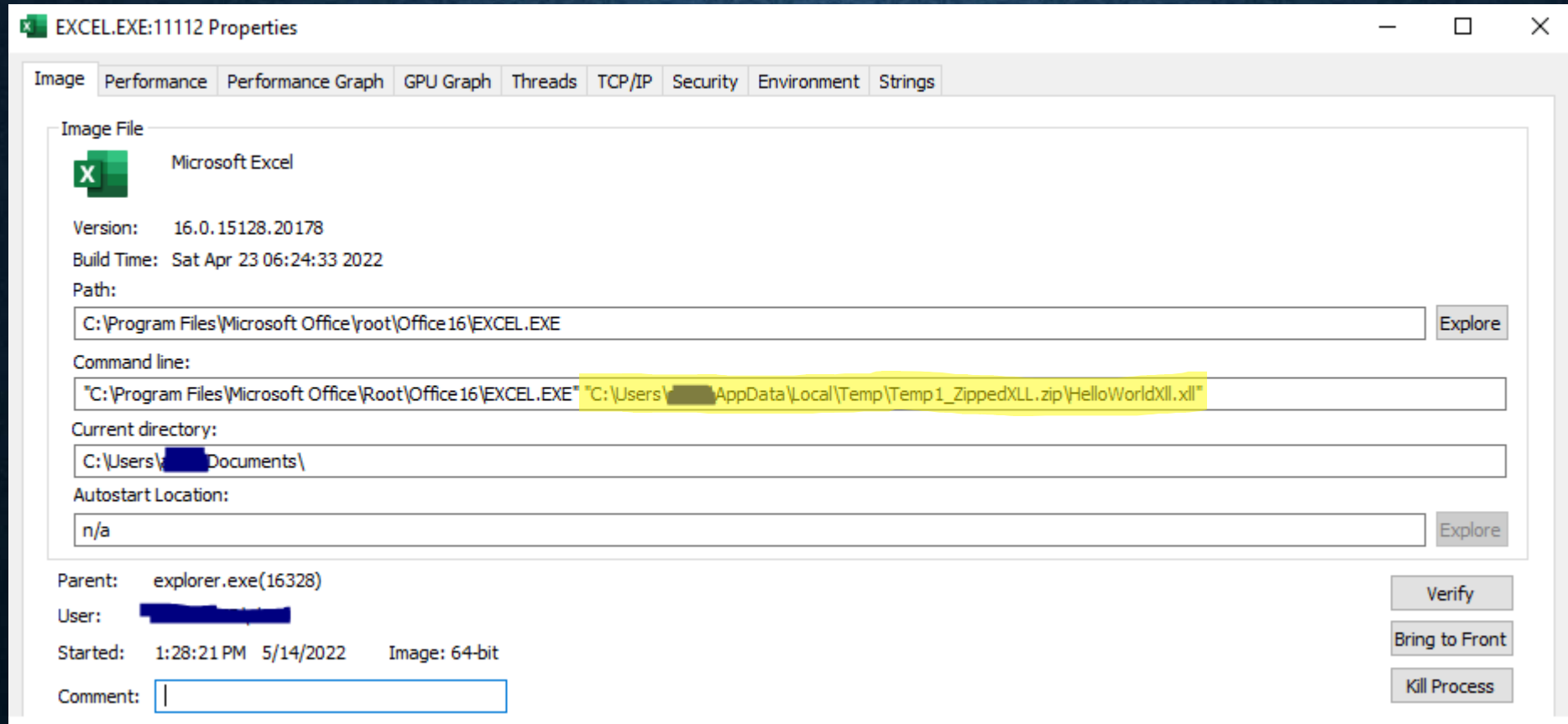
TROUBLE IN PARADISE



ZIP FILES AND EXECUTION

- Path of the “malicious” file is strange...
 - XLL was in c:\Users\user\Downloads\ZippedXLL.zip
 - Defender said it was
C:\Users\user\AppData\Local\Temp\Temp1_ZippedXLL.zip\HelloWorldXLL.xll
 - This is ZIP behavior, not XLL behavior:
 - When a file in a ZIP archive is opened without having been extracted, it is copied to
AppData\Local\Temp
 - Defender seems to have a signature for ANY code execution in the Temp folder... not a bad idea

ZIP FILES AND EXECUTION

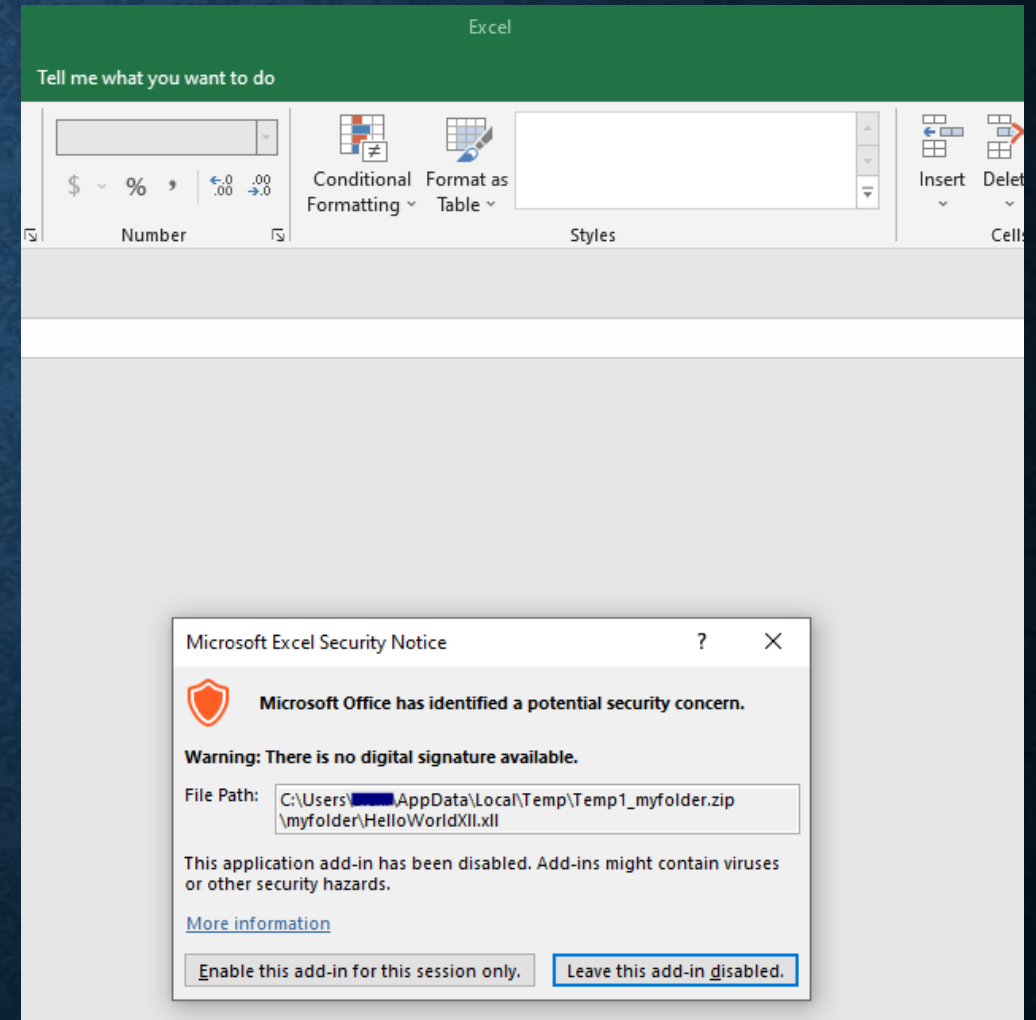
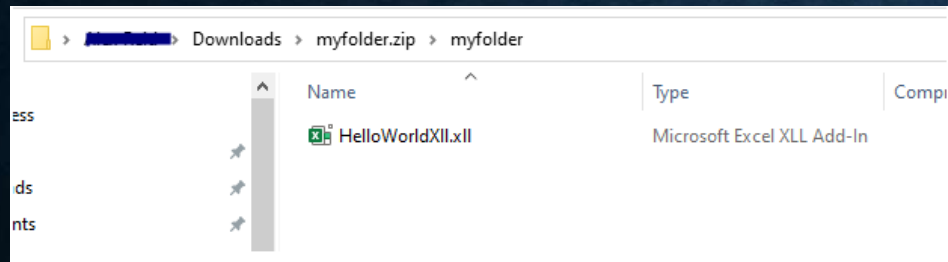
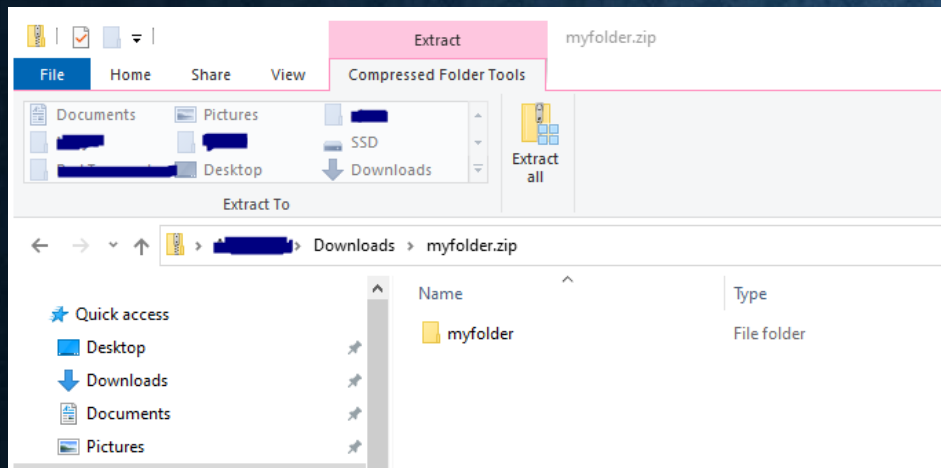


A ROCK AND A HARD PLACE...

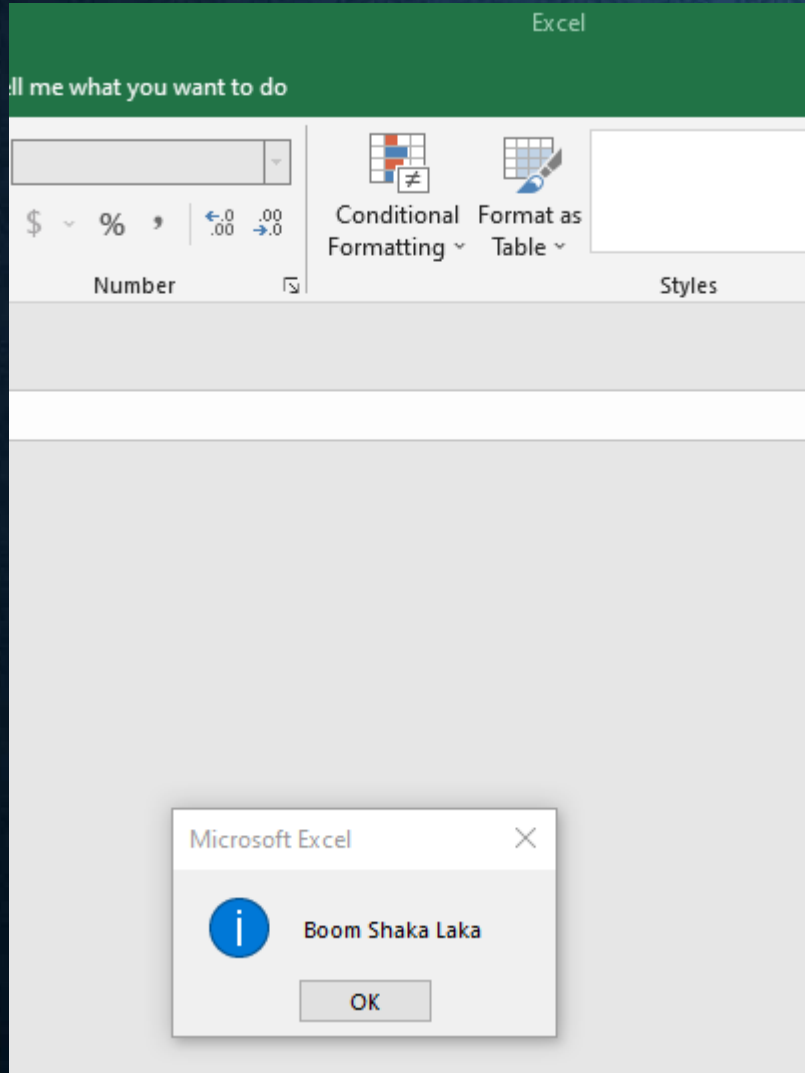
- ZIP's allow victims to download XLL from web
 - Executing XLL from within ZIP copies it to AppData\Local\Temp + flags AV
 - XLL executes without flagging AV if extracted first
 - Can we rely on victim to do so?
- Again other containers (ISO, IMG) won't work due to DLP
- How can we utilize ZIP's but avoid Defender flagging execution of the XLL?
 - Remember earlier about poking and prodding?

ONE STEP TO THE RIGHT

- What if we stick the XLL inside a folder?



SUCCESS!



TRADECRAFT

- We have code execution... now what?
- Goals:
 - Run our implant
 - Avoid suspicion from the victim
 - Clean up the XLL



EXECUTING THE IMPLANT

- To reiterate, XLL's are C/C++; SO much you can do when it comes to malware
- Utilize all the standard tips and tricks
 - Encode/Encrypt shellcode within the XLL
 - Sandbox detection/evasion
 - API hashing, direct syscalls, refresh NTDLL, etc.
 - Scope checks to ensure XLL only executes within target network (Domain, IP, etc)
- **Critically:**
 - Utilize an execution method that will run your implant in another process!
 - Victim can close Excel immediately after it opens and we are ok!

TRICKING THE VICTIM


- Pretexting led the victim to download and run the XLL
 - “Please see the spreadsheet containing...”
- Even after execution we prefer that the victim doesn’t suspect anything
- Embed an XLSX as a byte-array in our XLL
 - When the XLL executes, it runs our implant and then drops the XLSX to disk
 - Call `CreateProcess` on Excel + the XLSX and it opens to the screen

CLEANING UP


- If possible, prevent re-infection and remove XLL to make incident response more difficult / protect tooling
- Great project from LloydLabs: <https://github.com/LloydLabs/delete-self-poc>
 - Allows a running program to delete itself from disk; usually impossible because file is locked.
- Can utilize this code to delete XLL (AppData\Local\Temp) AND the original ZIP
- In addition to embedded XLSX, embed a new ZIP containing the XLSX

Before:

This PC > Downloads


Name	Date modified	Type	Size
▼ Today (1)			
 test.zip	10/17/2022 6:39 PM	Compressed (zipp...	359 KB

This PC > Downloads > test.zip > test


Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
 test.xll	Microsoft Excel XLL Add-In	359 KB	No	498 KB	29%	10/17/2022 6:39 PM

After:


This PC > Downloads

Name	Date modified	Type	Size
▼ Today (1)			
 test.zip	10/17/2022 6:44 PM	Compressed (zipp...	6 KB

This PC > Downloads > test.zip > test

Name	Type	Compressed size	Password ...	Size	Ratio	Date modified
 test.xlsx	Microsoft Excel Worksheet	6 KB	No	9 KB	30%	10/17/2022 6:39 PM

This PC > Local Disk (C:) > Users > █████ > AppData > Local > Temp > Temp5_test.zip > test

Name	Date modified	Type	Size
 test.xlsx	10/17/2022 6:44 PM	Microsoft Excel W...	9 KB

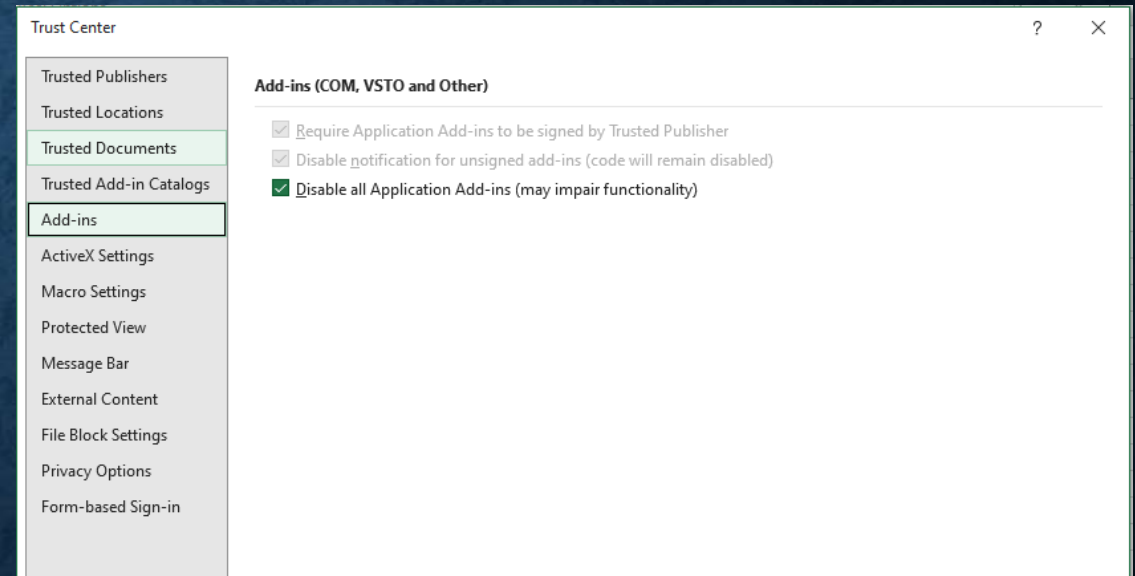
DEMO TIME!



MITIGATION

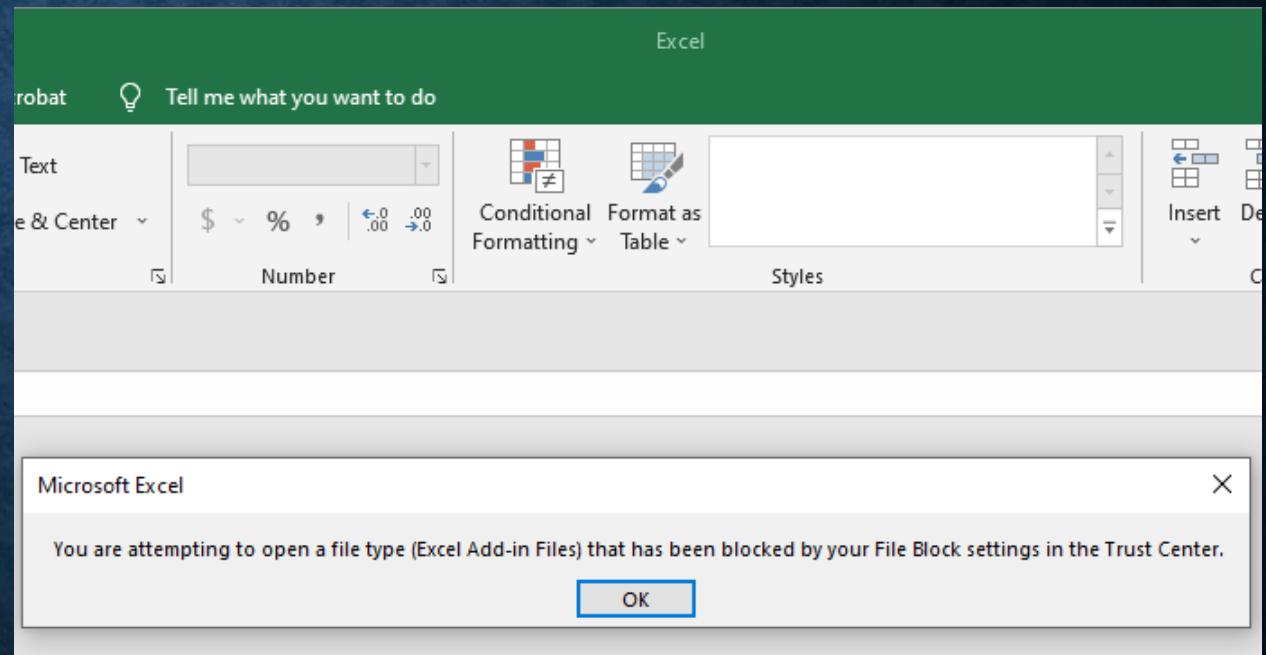
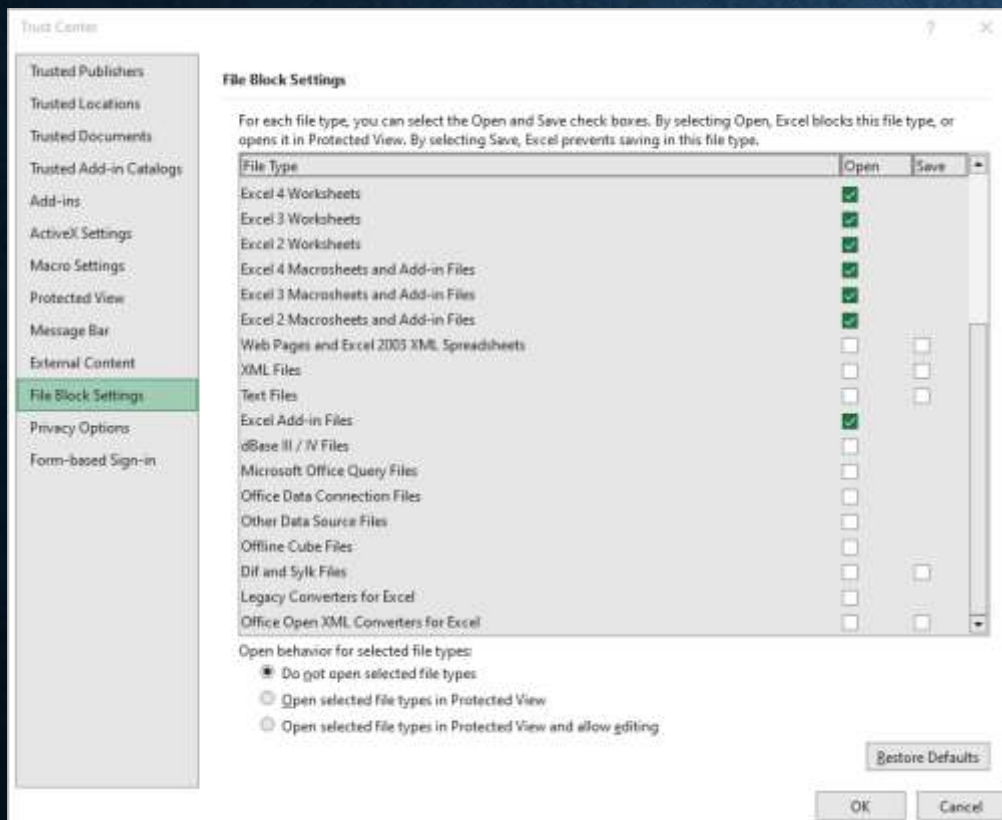
- How can organizations protect themselves from XLL's?
- Excel Trust Center
- While this sure looks like it should stop XLL's (Add-ins) from running, it doesn't!

This setting appears to relate to loading or registering Add-ins the intended way through Excel; double-clicking XLL's still works!



THE NUCLEAR OPTION

- Also in Excel Trust Center
- Utilizing File Block Settings, XLL's will now fail to execute when double-clicked



EXCEPT...

- Trust Center apparently doesn't apply to Office called via automation...
- Using COM objects we can still execute XLL's!

Demo Time!

QUESTIONS?

- Resources:
 - HelloWorldXll by edparcell: <https://github.com/edparcell/HelloWorldXll>
 - PackMyPayload by mgeeky: <https://github.com/mgeeky/PackMyPayload>
 - delete-self-poc by LloydLabs: <https://github.com/LloydLabs/delete-self-poc>
- Full write-up available at: https://github.com/Octoberfest7/XLL_Phishing