Patch Compliance On Windows

Patch Information
```
Get-Hotfix
```

Patch-Velocity
Counts the number of patches applied per day

```
Get-Hotfix | Sort-Object InstalledOn -Descending
```

Patch-Age
Patch age of a system is the number of days since the last patch was applied:

```
$lastPatchDate = (Get-HotFix | Sort-Object InstalledOn -Descending |
Select-Object -First 1).InstalledOn
$lastPatchDate
(New-TimeSpan -Start $lastPatchDate -End (Get-Date)).TotalDays
```

Patch Compliance on Debian-based Linux Distributions

Change shell to Powershell Core
```
Pwsh
```

Patches installed by the Apt package manager are logged in:
/var/log/dkpg.log

Patch-Velocity
Counts the number of patches applied per day
```
Get-Content /var/log/dpkg.log* | Select-String “ install “ -NoEmphasis
Get-Content /var/log/dpkg.log* |
   Select-String " install " -NoEmphasis |
   Out-File ./patches.txt -Encoding ascii
$lines = Get-Content ./patches.txt
($lines | Where-Object { $_ -match "^[0-9]" }) -replace " .*$"
```

Patch-Age
Patch age of a system is the number of days since the last patch was applied:
```
$lastPatchDate = ($lines |
   Where-Object { $_ -match "^[0-9]" }) -replace " .*$"  |
   Select-Object -last 1
$patchAge = (New-TimeSpan -Start (Get-Date -date $lastPatchDate) `
   -End (Get-Date)).TotalDays
"Last Patch Date: $lastPatchDate"
"Patch Age: $patchAge"
```

Windows Compliance Measurements

Check that the Administrator account is disabled:
```
Get-LocalUser -Name Administrator
```

Check that the Guest account is disabled:
```
Get-LocalUser -Name Guest
```

Save the list of local users to a variable and then test to see if both Guest and Administrator are disabled:
```
$disabledUsers = Get-LocalUser | Where-Object Enabled -eq $False
($disabledUsers.Name -contains 'Administrator') -And
($disabledUsers.Name -contains 'Guest')
```

Enumerate the members of local groups:
```
(Get-LocalGroupMember -Name Administrators | Measure-
Object).Count
(Get-LocalGroupMember -Name 'Power Users').Count
```

Check that a Windows services is installed, enabled and running:
```
((Get-Service -Name '<service name>').Count -ge 1 ) -And
((Get-Service -Name '<service name>').Status -eq 'running') -And
((Get-Service -Name '<service name>').StartType -like 'Automatic*')
```

All commands, unless stated otherwise, have been tested in the **SEC557: Continuous Automation for Enterprise and Cloud Compliance** course VMs using PowerShell Core.

SANS Cybersecurity Leadership Curriculum

SANS
Cybersecurity Leadership

RESOURCES

- sans.org/cybersecurity-leadership
- SANS Security Leadership
- @secleadership
- Recommended Reading
- Webcasts
- Blogs

AUD507: Auditing & Monitoring Networks, Perimeters & Systems
Auditing a security program and controls

LEG523: Law of Data Security and Investigation
Understanding legal and regulatory requirements

MGT414: SANS Training Program for the CISSP® Certification
Need Training for the CISSP® Exam? Here It Is.

MGT415: A Practical Introduction to Cyber Security Risk Management
Understanding security risk management

MGT433: Managing Human Risk: Mature Security Awareness Programs
Building & leading a security awareness program

MGT512: Security Leadership Essentials for Managers
Leading security initiatives to manage information risk

MGT514: Security Strategic Planning, Policy, and Leadership
Aligning security initiatives with strategy

MGT516: Managing Security Vulnerabilities: Enterprise & Cloud
Building & leading a vulnerability management program

MGT520: Leading Cloud Security Design & Implementation
Building and leading a cloud security program

MGT521: Leading Cybersecurity Change: Building A Security-Based Culture
Leading and aligning security initiatives with culture

MGT525: IT Project Management & Effective Communication
Managing security initiatives and projects

MGT551: Building and Leading Security Operations Center
Building and leading a security operations center

SEC440: CIS Critical Controls: A Practical Introduction
Introduction to CIS Critical Security

SEC474: Building A Healthcare Security & Compliance Program
You Must Do Compliance. Make it Valuable.

SEC557: Continuous Automation for Enterprise & Cloud Compliance
Measure What Matters - Not What's Easy

SEC566: Implementing and Auditing CIS Critical Controls
Building and auditing CIS Critical Controls

SANS
CLOUD
SECURITY

POWERSHELL FOR ENTERPRISE AND CLOUD COMPLIANCE

By AJ Yawn
Cheat Sheet v1.0.0

SANS.ORG/CLOUD-SECURITY

SANS.ORG/SEC557

SANS Cloud Security Curriculum

SANS
CLOUD
SECURITY

RESOURCES

- sans.org/cloud-security
- SANS Cloud Security
- @SANSCloudSec
- SANS Cloud Security
- Webcasts
- Blogs

SEC488: Cloud Security Essentials
License To Learn Cloud Security

SEC510: Public Cloud Security: AWS, Azure, and GCP
Multiple Clouds Require Multiple Solutions

SEC522: Defending Web Applications Security Essentials
Not a matter of "if" but "when". Be prepared for a web app attack. We'll teach you how.

SEC534: Secure DevOps: A Practical Introduction
Principles! Practices! Tools! Oh My! Start your journey on the DevSecOps road here.

SEC540: Cloud Security and DevSecOps Automation
The cloud moves fast. Automate to keep up.

SEC541: Cloud Monitoring and Threat Detection
Attackers can run, but not hide! Our radar sees all threats.

SEC557: Continuous Automation for Enterprise and Cloud Compliance
Using Cloud and DevOps Tools to Measure Security and Compliance

SEC584: Cloud Native Security: Defending Containers and Kubernetes
Deploy Scurely at the Speed of Cloud Native

SEC588: Cloud Penetration Testing
Aim your arrows to the sky and penetrate the Cloud.

FOR509: Enterprise Cloud Forensics and Incident Response
Find the Storm in the Cloud

MGT516: Managing Security Vulnerabilities: Enterprise & Cloud
Stop treating the symptoms. Cure the disease.

MGT520: Leading Cloud Security Design & Implementation
Building and leading a cloud security program

Review our Job Role Flight Plan at sans.org/cloud-security

AWS Compliance Measurements

Make sure current version AWS PowerShell module is available for use.

```
Install-Module -name AWSPowerShell.NetCore -Scope CurrentUser -Force
```

Load AWS Module

```
Import-Module AWSPowerShell.NetCore
```

Authenticate to AWS

```
Set-AWSCredential -StoreAs <name of profile>  -AccessKey YourAccessKeyHere -SecretKey YourSecretKeyHere
```

CIS AWS Benchmark Control 1.4

```
(Get-IAMAccountSummary).AccountAccessKeysPresent
```

CIS AWS Benchmark Control 1.5

```
(Get-IAMAccountSummary).AccountMFAEnabled
```

CIS AWS Benchmark Control 1.8

```
Get-IAMAccountPasswordPolicy
```

CIS AWS Benchmark Control 1.13

```
Get-IAMUserList | ForEach-Object {  Get-IAMAccessKey -UserName $_.UserName }
```

CIS AWS Benchmark Control 1.15

```
(Get-IAMUserList | ForEach-Object  {
   Get-IAMUserPolicies -UserName $_.UserName | Select-Object PolicyName
   Get-IAMAttachedUserPolicies -UserName $_.UserName | Select-Object PolicyName
}
```

CIS AWS Benchmark Control 3.1

```
Get-CTTrail
```

Measure VMWare host configuration

Gather information about the VMWare host system and configuration

```
Get-VMHost -Server <name>
```

Validate common hypervisor settings

```
(Get-VMHost).ExtensionData
```

Leverage the Config property in ExtensionData to get in depth config settings (example DNS resolver configuration below)

```
(Get-VMHost).ExtensionData.Config.Network.DNSConfig
```

Measure if DNS settings are configured correctly:

```
$dnsservers = (Get-VMHost).ExtensionData.Config.Network.DNSConfig | Select-Object -ExpandProperty address
$dnsservers -contains '8.8.8.8'
$dnsservers -contains '8.8.4.4'
```

Validate the NTP server(s) configured on VMWare host

```
Get-VMHost -Server <name> | Get-VMHostNtpServer
```

Validate that the NTP service is running and is configured to run at startup

```
Get-VMHost | Get-VMHostService | Where-Object {$_.key -eq "ntpd"} | Select-Object VMHost, Label, Key, Policy, Running, Required
```

Patch Data

```
(Get-ESXCli -Server esxi1).software.vib.list()
```

Patch velocity

```
(Get-ESXCli -Server esxi1).software.vib.list() | Group-Object InstallDate
```

Patch Age

```
$lastPatchDate = ((Get-ESXCli -Server esxi1).software.vib.list() | Sort-Object InstallDate -Descending | Select-Object -First 1).InstallDate
$patchAge = (New-TimeSpan -Start $lastPatchDate -End (Get-Date)).TotalDays
$patchAge
```

Review Nessus Vulnerability Scan Details

Navigate and set the location of the Nessus files

```
Set-Location C:\user\Desktop\2021Scans
```

View what files exist in the directory

```
Get-ChildItem
```

Let's assume there are a lot of Nessus files to process, save them to a variable

```
$scanResults = Import-Csv -path (Get-ChildItem *.csv |
  Select-Object -ExpandProperty FullName)
```

Group the results by Risk

```
$scanResults | Group-Object Risk
$scanResults | Group-Object Risk | Where-Object Name -eq 'Critical'
```

Identify hosts with largest numbers of critical vulnerabilities

```
$scanResults |
  Where-Object Risk -eq 'critical' |
  Group-Object Host |
  Select-Object Count, Name |
  Where-Object Count -gt 5 |
  Sort-Object Count -Descending
```

Identify percent of vulnerabilities marked as critical

```
$criticalCount =
  ($scanResults |
    Group-Object Risk |
      Where-Object Name -eq 'Critical'
  ).Count
$totalCount = ($scanResults |  Where-Object Risk -ne 'None').Count
$criticalCount/$totalCount
```