**Patch Development - PowerShell**

We worked on creating PowerShell scripts for:

1. Tenable Software Update Script: This script checks for updates, downloads the latest version, and installs it.
2. General Vulnerability Patching Script: This script applies patches to software using a predefined list of patches.
3. Combining Scripts: We combined the Tenable update and general patching scripts.
4. Patch Management Script: This script defines a list of patches, downloads, and installs them on multiple target computers.
5. Pushing updates to multiple PCs using a downloaded MSI file.

**Script 1: Local Patching Script**

This script will download and install patches locally on the machine it's run on.

PowerShell

# Define patch list with software name, download URL, and installation command

$patches = @(

@{

Name = "Tenable Nessus"

Url = "https://www.tenable.com/downloads/nessus"

Msi = "msiexec.exe /i Nessus.msi /quiet /norestart"

},

@{

Name = "Adobe Acrobat"

Url = "https://www.adobe.com/support/downloads/detail.html?ftpID=XXXXX"

Exe = "AdobeAcrobat.exe /s /norestart"

},

@{

Name = "Microsoft KB456789"

Url = "https://www.microsoft.com/en-us/download/details.aspx?id=XXXXX"

Msi = "msiexec.exe /i KB456789.msi /quiet /norestart"

}

)

# Loop through patches

foreach ($patch in $patches) {

Write-Host "Applying patch: $($patch.Name)"

# Download patch

$downloadPath = "$env:TEMP\$($patch.Name).msi"

Invoke-WebRequest -Uri $patch.Url -OutFile $downloadPath -UseBasicParsing

# Install patch

if ($patch.Msi) {

Start-Process -FilePath "msiexec.exe" -ArgumentList "/i $downloadPath /quiet /norestart" -Wait

} elseif ($patch.Exe) {

Start-Process -FilePath $downloadPath -ArgumentList $patch.Exe -Wait

}

}

**Script 2: Batch Patching Script for Multiple Computers**

This script will push updates to multiple computers at the same time using a downloaded MSI file.

PowerShell

# Define text file path containing PC information

$pcListPath = "C:\PCs.txt"

# Define .msi file path

$msiFilePath = "C:\path\to\update.msi"

# Define credentials for remote access

# If you are **using Elevated Admin Account environment**, you don’t need this credentials script

$username = "Domain\Username"

$password = ConvertTo-SecureString "Password" -AsPlainText -Force

# “cred” variable stores a credentials info.

$cred = New-Object System.Management.Automation.PSCredential($username, $password)

# Read PC list from text file

$pcs = Get-Content -Path $pcListPath

# Loop through PCs

foreach ($pc in $pcs) {

Copy-Item -Path $msiFilePath -Destination "\\$pc\C$\Temp"

Invoke-Command -ComputerName $pc -Credential $cred -ScriptBlock {

param ($msiFilePath)

Start-Process -FilePath "msiexec.exe" -ArgumentList "/i C:\Temp\$(Split-Path -Leaf $msiFilePath) /quiet /norestart" -Wait

} -ArgumentList $msiFilePath

}

**Note:**

* Make sure to update the $patches array in the Local Patching Script with the software name, download URL, and installation command.
* Update the $pcListPath and $msiFilePath variables in the Batch Patching Script with the correct paths.
* Ensure you have the necessary permissions and access rights to run these scripts.
* Test the scripts in a non-production environment before running them in production.
* Verify patch installation on target computers.

**Script 3:**

**Batch Patching with Timeout Limit, Writing Log Report, Status Update and deleting the EXE file.**

This script will push updates to multiple computers at the same time using downloaded EXE file and displays status/prints the log report.

<#This is Batch Patching Script which uses ".exe" file. This script also writes the log report. This script uses "-Timeout" parameter of max 300 sec

for any PC to install the update, after that the update process will fail and move to next PC#>

# Define text file path containing PC information

$pcListPath = "C:\PCs.txt"

# Define .exe file path

$exeFilePath = "C:\path\to\npp.8.6.7.Installer.x64.exe"

# Read PC list from text file

$pcs = Get-Content -Path $pcListPath

# Create a log file to store the status updates

$logFilePath = "C:\PatchInstallationLog.txt"

# Define the timeout period in seconds

$timeoutSeconds = 300 # 5 minutes

# Loop through PCs

foreach ($pc in $pcs) {

Write-Host "Installing patch on $pc..."

Copy-Item -Path $exeFilePath -Destination "\\$pc\C$\Temp"

Invoke-Command -ComputerName $pc -ScriptBlock {

param ($exeFilePath)

$installationResult = Start-Process -FilePath "C:\Temp\$(Split-Path -Leaf $exeFilePath)" -ArgumentList "/S" -Wait -PassThru

if ($installationResult.ExitCode -eq 0) {

Write-Host "Patch installation successful on $env:COMPUTERNAME"

"Patch installation successful on $env:COMPUTERNAME" | Add-Content -Path $logFilePath

} else {

Write-Host "Patch installation failed on $env:COMPUTERNAME with exit code $($installationResult.ExitCode)"

"Patch installation failed on $env:COMPUTERNAME with exit code $($installationResult.ExitCode)" | Add-Content -Path $logFilePath

}

# Delete the .exe file after installation

Remove-Item -Path "C:\Temp\$(Split-Path -Leaf $exeFilePath)" -Force

} -ArgumentList $exeFilePath -Timeout $timeoutSeconds

}

Above script removes the EXE file from each PC after each installation complete, if I want to start the delete EXE file after the completion of entire installation process then the script must be outside the “foreach” loop, such as below.

# Loop through PCs

foreach ($pc in $pcs) {

Write-Host "Installing patch on $pc..."

Copy-Item -Path $exeFilePath -Destination "\\$pc\C$\Temp"

Invoke-Command -ComputerName $pc -ScriptBlock {

param ($exeFilePath)

$installationResult = Start-Process -FilePath "C:\Temp\$(Split-Path -Leaf $exeFilePath)" -ArgumentList "/S" -Wait -PassThru

if ($installationResult.ExitCode -eq 0) {

Write-Host "Patch installation successful on $env:COMPUTERNAME"

"Patch installation successful on $env:COMPUTERNAME" | Add-Content -Path $logFilePath

} else {

Write-Host "Patch installation failed on $env:COMPUTERNAME with exit code $($installationResult.ExitCode)"

"Patch installation failed on $env:COMPUTERNAME with exit code $($installationResult.ExitCode)" | Add-Content -Path $logFilePath

}

} -ArgumentList $exeFilePath -Timeout $timeoutSeconds

}

# Delete the .exe file on each PC after the entire list of PCs has been updated

foreach ($pc in $pcs) {

Invoke-Command -ComputerName $pc -ScriptBlock {

Remove-Item -Path "C:\Temp\npp.8.6.7.Installer.x64.exe" -Force

}

}