Windows Operations Plan

Welcome to the Windows Ops Plan!

This plan is intended to be a guide to windows operations and procedures for CCDC. This plan encompasses several key areas that are essential for the success of Windows incident response during the competition.

Resources

Initial Management Configs

There are several tools and configs that must be applied to support better management of the Windows systems by team members and Serial Scripter.

- [x] ~~PowerShell Update~~
- [x] ~~SSH~~
- [ ] Logbeat
- [ ] Sysmon
- [ ] AV

PowerShell Update

Updating PowerShell is essential on machines older than Windows 10 era operating systems. Many of them run version older than 5.1 which is important for running other scripts. There is an upgrade script, **BUT** it does not always work in which case you must know how to do this operation manually.

Method 1: Use the Script

The below script attempts to automatically install PowerShell 5.1 on a given machine. It support Windows 7, Windows Server 2008 (sometimes), Windows 8, and Windows Server 2012. Reason this script may not work include:

- TLS version mismatch
- Internet connectivity error
- Missing cmdlets or .NET objects missing on the host computer

- Microsoft URL 404

That last error is actually easily fixable and you should attempt to fix the script if it happens as it will still be faster than manually installing the update. All you need to do is grab the new download link for the WMF package you need and put it in the correct URL variable in the script `$url`

https://github.com/Hunter-Pittman/windows-belt/blob/main/install51.ps1

```
$osVersion = [System.Environment]::OSVersion.Version
if (($osVersion.Major -eq 6 -and $osVersion.Minor -eq 1) -or ($osVers
    $psVersion = $PSVersionTable.PSVersion.major
    if ($psVersion -lt "5") {
        Write-Output "Installing PowerShell 5.1..."

        # Download the PowerShell 5.1 installer
        if ($osVersion.Major -eq 6 -and $osVersion.Minor -eq 1) {
            $url = "https://download.microsoft.com/download/3/0/D/30D
        } elseif ($osVersion.Major -eq 6 -and $osVersion.Minor -eq 2)
            $url = "https://download.microsoft.com/download/3/0/D/30D
        } else {
            $url = "https://download.microsoft.com/download/3/0/D/30D
        }
        $zipfile = "$env:TEMP\PowerShell5.1.zip"
        $webclient = New-Object System.Net.WebClient
        $webclient.DownloadFile($url, $zipfile)

        # Extract the installer files
        $extractPath = "$env:TEMP\PowerShell5.1"
        if (!(Test-Path $extractPath -PathType Container)) {
            New-Item -ItemType Directory -Path $extractPath | Out-Nul
        }

        $shellApplication = New-Object -ComObject Shell.Application
        $zipPackage = $shellApplication.NameSpace($zipfile)
        $destinationFolder = $shellApplication.NameSpace($extractPath
        $destinationFolder.CopyHere($zipPackage.Items(), 0x14)

        # Install PowerShell 5.1
        $msuFile = Get-ChildItem -Path $extractPath -Filter '*.msu' |
        wusa.exe $msuFile.FullName /quiet /norestart

        Write-Output "PowerShell 5.1 installed."
    } else {
        Write-Output "PowerShell 5.1 is already installed."
    }
} else {
    Write-Output "This script is only supported on Windows 7, 8, and
}
```

💡 NOTE: Above is the code as of 11/27/23. This is to give context for this section, please use the latest code from GitHub.

After the script is run successfully **RESTART THE COMPUTER f**or the update to take effect.

Method 2: Install 5.1 Manually

If the script does not work you can install 5.1 manually download WMF. Below is the link to the WMF download page. Alternatively you can download the WMF you need via link for the download which will also be listed below.

WMF Download Links

Download WMF 5.1 from Official Microsoft Download Center

Windows Management Framework 5.1 includes updates to Windows PowerShell, Windows PowerShell Desired State Configuration (DSC), Windows Remote Management (WinRM), Windows Management Instrumentation (WMI). Release notes: https://go.microsoft.com/fwlink/?linkid=839460

https://www.microsoft.com/en-us/download/details.aspx?id=54616

▼ Individual Download Links

Windows 7 and Server 2008: https://download.microsoft.com/download/3/0/D/30DB904F-E9EB-4C22-9630-A63A84FD7E1D/Win7AndW2K8R2-KB3191566-x64.zip

Windows Server 2012 R2: https://download.microsoft.com/download/3/0/D/30DB904F-E9EB-4C22-9630-A63A84FD7E1D/W2K12-KB3191566-x64.zip

Windows Server 2012 R1 (sometimes) and Windows 2008: https://download.microsoft.com/download/3/0/D/30DB904F-E9EB-4C22-9630-A63A84FD7E1D/W2K12R2-KB3191565-x64.zip

Procedure

1. Determine the PowerShell version: `PS C:\> $PSVersionTable`
2. If your version is below 5.1 you need to attempt an upgrade
3. Determine you OS version: `systeminfo | findstr /B /C:"OS Name" /B /C:"OS Version"`
4. Pick the appropriate WMF download link from above or visit the download portal and get the appropriate package from their.
5. Extract the download zip file
6. Inside the folder you will find a PowerShell script and a .msu file. You can delete the PowerShell script
7. Open a new terminal in the directory you extracted the .msu too.
8. Run the following command to install the PowerShell update `wusa.exe [NAME OF MSU FILE]`
9. A dialog should pop up letting you know the update status
10. If this did not work for whatever reason, attempt to run the .msu file through double clicking and troubleshoot from there
11. Finally restart the system so the update will be applied

SSH Configuration

SSH configuration on Windows is awful, at least for consistent installs across distributions. Like the PowerShell update section there is a script to do it, but it doesn't quite work the same on all distributions and additional configuration maybe required.

```
https://raw.githubusercontent.com/Hunter-Pittman/windows-belt/main/ssh_setup.ps1
```

```
param (
    [string]$PublicKey
)

$sshConfig = @"
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# The strategy used for options in the default sshd_config shipped wi
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override t
# default value.

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey __PROGRAMDATA__/ssh/ssh_host_rsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_dsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_ecdsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

```
#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authoriz
# but this is overridden so installations will only check .ssh/author
AuthorizedKeysFile  .ssh/authorized_keys

#AuthorizedPrincipalsFile none

# For this to work you will also need host keys in %programData%/ssh/
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#PermitUserEnvironment no
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
```

```
#Banner none

# override default of no subsystems
Subsystem   sftp    sftp-server.exe

# Example of overriding settings on a per-user basis
#Match User anoncvs
#   AllowTcpForwarding no
#   PermitTTY no
#   ForceCommand cvs server

#Match Group administrators
#       AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authori

PubkeyAuthentication yes
AuthorizedKeysFile .ssh/authorized_keys

"@

function SSH-Setup {
    ## Set network connection protocol to TLS 1.2
    [Net.ServicePointManager]::SecurityProtocol = [Net.SecurityProtoc
    #[Net.ServicePointManager]::SecurityProtocol = [Net.ServicePointM
    #[System.Net.ServicePointManager]::ServerCertificateValidationCal

    ## Define the OpenSSH latest release url
    $url = 'https://github.com/PowerShell/Win32-OpenSSH/releases/late
    ## Create a web request to retrieve the latest release download l
    $request = [System.Net.WebRequest]::Create($url)
    $request.AllowAutoRedirect=$false
    $response=$request.GetResponse()
    $source = $([String]$response.GetResponseHeader("Location")).Repl
    ## Download the latest OpenSSH for Windows package to the current
    $webClient = [System.Net.WebClient]::new()
    $webClient.DownloadFile($source, (Get-Location).Path + '\OpenSSH-

    # Extract the ZIP to a temporary location
    Expand-Archive -Path .\OpenSSH-Win64.zip -DestinationPath ($env:t
    # Move the extracted ZIP contents from the temporary location to
    Move-Item "$($env:temp)\OpenSSH-Win64" -Destination "C:\Program F
    # Unblock the files in C:\Program Files\OpenSSH\
    Get-ChildItem -Path "C:\Program Files\OpenSSH\" | Unblock-File

    & 'C:\Program Files\OpenSSH\install-sshd.ps1'

    ## changes the sshd service's startup type from manual to automat
    Set-Service sshd -StartupType Automatic
    ## starts the sshd service.
    Start-Service sshd

    try {
        New-NetFirewallRule -Name sshd -DisplayName 'Allow SSH' -Enab
    } catch {
        Write-Host "Firewall rule already exists"
    }

    try {
        New-ItemProperty -Path "HKLM:\SOFTWARE\OpenSSH" -Name Default
    } catch {
        Write-Host "DefaultShell registry key already exists or there
    }

    $InheritanceFlag = [System.Security.AccessControl.InheritanceFlag
    $PropagationFlag = [System.Security.AccessControl.PropagationFlag
    $objType = [System.Security.AccessControl.AccessControlType]::All

    $Path = "C:\Program Files\OpenSSH\"

    $acl = Get-Acl $Path
    $permission = "NT Authority\Authenticated Users","ReadAndExecute"
    $accessRule = New-Object System.Security.AccessControl.FileSystem

    $acl.SetAccessRule($accessRule)
    Set-Acl $Path $acl

    ## Configure the OpenSSH server to use public key authentication
    Set-Content -Path "C:\ProgramData\ssh\sshd_config" -Value $sshCon
    ## Add the provided public key to the server's authorized keys fi
    $authorizedKeysPath = "$env:USERPROFILE\.ssh\authorized_keys"
    if (Test-Path $authorizedKeysPath) {
        Add-Content -Path $authorizedKeysPath -Value $PublicKey
    } else {
        New-Item -ItemType Directory -path "$env:USERPROFILE\.ssh\"
        New-Item -ItemType File -Path $authorizedKeysPath
        Add-Content -Path $authorizedKeysPath -Value $PublicKey
    }

    Restart-Service sshd
}

SSH-Setup
```

💡 NOTE: Above is the code as of 11/27/23. This is to give context for this section, please use the latest code from GitHub.

How to Run

Run the above script by doing `./ssh_setup.ps1 -PublicKey [YOUR PUBLIC KEY HERE']`. The public key section is not necessarily essential if you don't wish to configure it right away.

Older Windows

The above code **DOES ALOT** and the majority of it should work on any Windows distribution, **HOWEVER** on older versions of Windows additional configuration maybe required in which case you will need to run `ssh_legacy_fix.ps1`.

https://raw.githubusercontent.com/Hunter-Pittman/windows-belt/main/ssh_legacy_fix.ps1

```
param (
    [string]$PublicKey
)

$sshConfig = @"
# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# The strategy used for options in the default sshd_config shipped wi
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override t
# default value.

#Port 22
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

#HostKey __PROGRAMDATA__/ssh/ssh_host_rsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_dsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_ecdsa_key
#HostKey __PROGRAMDATA__/ssh/ssh_host_ed25519_key

# Ciphers and keying
#RekeyLimit default none

# Logging
#SyslogFacility AUTH
#LogLevel INFO

# Authentication:

#LoginGraceTime 2m
#PermitRootLogin prohibit-password
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10

#PubkeyAuthentication yes

# The default is to check both .ssh/authorized_keys and .ssh/authoriz
# but this is overridden so installations will only check .ssh/author
#AuthorizedKeysFile .ssh/authorized_keys

#AuthorizedPrincipalsFile none

# For this to work you will also need host keys in %programData%/ssh/
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no

#AllowAgentForwarding yes
#AllowTcpForwarding yes
#GatewayPorts no
#PermitTTY yes
#PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#PermitUserEnvironment no
#ClientAliveInterval 0
#ClientAliveCountMax 3
#UseDNS no
#PidFile /var/run/sshd.pid
#MaxStartups 10:30:100
#PermitTunnel no
#ChrootDirectory none
#VersionAddendum none

# no default banner path
#Banner none

# override default of no subsystems
```

```
Subsystem   sftp    sftp-server.exe

# Example of overriding settings on a per-user basis
#Match User anoncvs
#   AllowTcpForwarding no
#   PermitTTY no
#   ForceCommand cvs server


PubkeyAuthentication yes
PasswordAuthentication no

Match Group administrators
       AuthorizedKeysFile __PROGRAMDATA__/ssh/administrators_authoriz

"@


function UTF8NoBom($filter) {
 $Utf8NoBomEncoding = New-Object System.Text.UTF8Encoding($False)
 foreach($i in ls -Recurse -Filter $filter) {
 $MyFile = Get-Content $i.fullname
 [System.IO.File]::WriteAllLines($i.fullname, $MyFile, $Utf8NoBomEnco
 }
}

function SSH-Fix {
    ## Configure the OpenSSH server to use public key authentication
    Set-Content -Path "C:\ProgramData\ssh\sshd_config" -Value $sshCon
    New-Item -Type File "C:\ProgramData\ssh\" -Name "administrators_a

    ## Add the provided public key to the server's authorized keys fi
    $authorizedKeysPath = "C:\ProgramData\ssh\administrators_authoriz
    if (Test-Path $authorizedKeysPath) {
        Add-Content -Path $authorizedKeysPath -Value $PublicKey
    } else {
        New-Item -ItemType Directory -path "C:\ProgramData\ssh\"
        New-Item -ItemType File -Path $authorizedKeysPath
        Add-Content -Path $authorizedKeysPath -Value $PublicKey
    }


    Set-Location “$ENV:ProgramData\ssh”
    UTF8NoBom(“*authorized_keys*”)

    $ak = “$ENV:ProgramData\ssh\administrators_authorized_keys”
    $acl = Get-Acl $ak
    $acl.SetAccessRuleProtection($true, $false)
    $administratorsRule = New-Object system.security.accesscontrol.fi
    $systemRule = New-Object system.security.accesscontrol.filesystem
    $acl.SetAccessRule($administratorsRule)
    $acl.SetAccessRule($systemRule)
    $acl | Set-Acl

}

SSH-Fix
```

💡 NOTE: Above is the code as of 11/27/23. This is to give context for this section, please use the latest code from GitHub.

How to Run

Just run `ssh_legacy_fix.ps1 -PublicKey [YOUR PUBLIC KEY]`. If your public key does not work add it manually to the `administrators_authorized_keys` file.

Troubleshooting

The main source of errors will be at the top of the program regarding TLS. Older Windows distributions can have issues downloading things from the internet without properly configuring TLS. The line currently uncommented for that section sees the most universal success but if you get TLS errors try some of the other lines.

Comment the currently uncommented line and uncomment a different one and rerun to see if it works. If this still doesn’t work take the openssh url https://github.com/PowerShell/Win32-OpenSSH/releases/latest/ download it manually and make sure it is in the same directory as the script. Rerun the script and wait for it to complete.

Assessment and Inventory

By the time you are ready to do the things in this section you should have done the initial management configs and have access to Serial Scripter as well as SSH and Ansible access. Some of these sections will utilize Ansible playbooks to acquire the needed information, but the manual way to acquire it will also be listed just in case.

Vulnerability Assessment

In this section the Windows Environment must be scanned using OpenVAS or an equivalent tool. The results of this process is essential for patching as many holes as possible in the security of the Windows environment.

Installed Apps

GET FUCKED

Group and Local Policy

EzScript or other automation will be taking care of this

Services

Ansible

Run the list_services.yml playbook.

Manual

In PowerShell run `Get-Service`

In CMD run `net start`

System Hardening and Security Baseline

EZScript

EZScript is a finicky script and should not operated by the untrained. It has the potential to completely wreck a system so leave it to Hunter until he can finish this part of the OPS manual

WinPEAS

WinPEAS, while it is malware, is a useful tool in determine potential attack vectors a red teamer might take. It is useful to run this script after initial configuration and recon is done.

Account Management

Active Directory

LDAP Queries:

- Get **Domain Admins:**

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<DOMAIN>\Administrator' -
  ```

  - If you want to get members of other groups, just switch the `CN=Domain Admins` to the group name that you want to query

- List all **groups** on a domain:

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<domain>\<user>' -w '<pas
  ```

  - The `cut` command just cuts out all the other BS to make the output easier to parse (;

- List all **users** on a domain:

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<domain>\<username>' -w '
  ```

- List all **administrators** on a domain:

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<domain>\<username>' -w '
  ```

- List all **remote desktop users**:

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<domain>\<username>' -w '
  ```

- List **group membership**:

  ```
  ldapsearch -x -H ldap://<DC IP ADDR> -D '<domain>\<user>' -w '<pas
  ```

NOTE: If we want to have a quick launch/cheatsheet integrated to our terminals to execute these types of commands, we can probably use Arsenal:

GitHub - Orange-Cyberdefense/arsenal: Arsenal is just a quick inventory and launcher for hacking programs

Arsenal is just a quick inventory and launcher for hacking programs - GitHub - Orange-Cyberdefense/arsenal: Arsenal is just a quick inventory and launcher for hacking programs

https://github.com/Orange-Cyberdefense/arsenal

AD Account Disable

```
Import-Module ActiveDirectory

# Get all AD user accounts
$users = Get-ADUser -Filter *

# Disable each user account
foreach ($user in $users) {
    Disable-ADAccount -Identity $user.SamAccountName
}
```

Local Accounts