Using PowerShell to Remotely Check the Windows FileWave Client Status

What

The FileWave Client on Windows is like any other software service...the service can be impacted by computer uptime, user interference, crashes, etc. This article gives you a way to INDEPENDENTLY check that a list of devices has the FileWave Client, and it is in working order (or not).

When/Why

Will you need this frequently? Unlikely, but all the same, it is a great tool for sweeping a network to look for devices and confirm the FileWave client (for Windows only). The code here does make some assumptions about your environment, but those are called out below.

How

So, you think the FileWave client may be broken or missing on some endpoints? Wouldn't it be great if you could verify that remotely rather than having to confirm the devices by hand. The following allows you to do just that.

```
#import a list of computers
$mypath=$MyInvocation.MyCommand.Path
$mypath=Split-Path $mypath -Parent
try {
    $computers=Get-Content $mypath\computers.txt -ErrorAction Stop
} catch {
    #no computers.txt file found
    write-host "`nTo use this utility, a text file called computers.txt must exist in the same location as the
script.  The file should contain one computer name or IP per line"
    break
}

foreach ($computer in $computers) {
    $online=$false
    try{
        #try to resolve the name
        $online = Resolve-DnsName $computer -quicktimeout -ErrorAction Stop
        $online = $true
    }catch{
        #Catching errors...machine offline
        $online= $false
    }

    if (!$online) {
        #device not online...show it in UI so that we see progress, but don't write it to the results file since
it isn't actionable
        write-host "$computer, Not online"
    } else {
        #device online, so let's just see if the service is there
        $fw_service=""
        try{
            #Getting service ...sometimes device might not allow collection (if RPC is unavailable for instance)
            $fw_service = Get-Service -ComputerName $computer  -Name 'FilewaveWinClient' -ErrorAction Stop
            $fw_service=$fw_service.Status
        }catch{
            #Catching errors...no filewave service
            $fw_service="no"
        }

        if ($fw_service -eq "no") {
            #no need to look further since we either can't talk to RPC, or there is no FW service
            write-host "$computer, No FW Service or RPC unavailable"
            Add-Content -Path $mypath\output.txt -Value "$computer, FW Needs Installed or RPC Unavailable"
```

```
        } else {
            #fw is there as a service, so let's return status, version, and server address
            try {
                #using C$ share, which won't require winrm
                $TargetPath = "\\$computer\C$\Program Files (x86)\FileWave\fwcld.exe"
                $fw_version = [System.Diagnostics.FileVersionInfo]::GetVersionInfo($TargetPath)
                $fw_version = $fw_version.ProductVersion
            } catch {
                #Catching errors
                $fw_version="version not readable"
            }

            #get fw server address from registry
            try {
                #read server address from registry
                #we need remote registry turned on to read, but we'll turn it back off
                #note this does not account for an environment where remote-registry is on by default...if so,
comment out the remote registry lines
                Get-Service -ComputerName $computer -Name RemoteRegistry | Set-Service -StartupType Manual -
PassThru| Start-Service

                $Reg = [Microsoft.Win32.RegistryKey]::OpenRemoteBaseKey('LocalMachine', $computer)
                $RegKey= $Reg.OpenSubKey("SOFTWARE\\WOW6432Node\\FileWave\\WinClient")
                $fw_server = $RegKey.GetValue("Server")

                #turn remote registry back off
                Get-Service -ComputerName $computer -Name RemoteRegistry | Set-Service -StartupType Disabled -
PassThru| Stop-Service

            } catch {
                #Catching errors...no registry
                $fw_server="server address not readable"
            }

            #write the output
            write-host "$computer, $fw_service, $fw_version, $fw_server"
            Add-Content -Path $mypath\output.txt -Value "$computer, $fw_service, $fw_version, $fw_server"

        }
    }
}
```

Assumptions made in the above code:
1. There is a text file called computers.txt in the same location as the Powershell script
2. That computers.txt file contains a computer name or IP per line (name is better if you have dynamic DNS)
3. That the Powershell itself is running from a Domain Admin account...this avoids any credential related issues
4. It is assumed that WinRM is not enabled in your environment (if it is this code could easily be made more elegant)

Note that this script could easily be modified to look at other services, to authenticate differently, and to take remediation. As provided, it simply provides a list of results of device name, FW service status, FW client version, and FW server address assigned. All very useful information for troubleshooting. PSEXEC is highly recommended for taking corrective action.

Related Content

- Script Best Practices
- Using PsExec to Remotely Restart the FileWaveWinClient Service
- FileWave Client Status Check: How to ask the client what it is doing on macOS and Windows

Digging Deeper

If you want a resource to pre-sweep the device list for devices that are online separately, you can use the following. A refined list will just make the above script run a bit faster.

```
#Let's just look for a list of devices online

#import a list of computers
$mypath=$MyInvocation.MyCommand.Path
```

```
$mypath=Split-Path $mypath -Parent
$computers=Get-Content $mypath\online_test.txt

foreach ($computer in $computers) {
    $online=$false
    try{
        #try to resolve
        $online = Resolve-DnsName $computer -quicktimeout -ErrorAction Stop
        $online = $true
        write-host $computer
    }catch{
        #Catching errors...machine offline
        $online= $false
    }
}
```

Revision #7
★Created 16 April 2024 09:32:55 by Tony Keller
✎Updated 19 September 2024 17:11:06 by Josh Levitsky