To create a **compiled PowerShell script** that accepts parameters for the server and port range, and can be called like a program (e.g., FindSqlPort.exe), follow these steps:

**✅ 1. PowerShell Script with Parameters**

Save this as FindSqlPort.ps1:

powershell

CopyEdit

param (

[string]$Server,

[int]$StartPort = 1433,

[int]$EndPort = 1445

)

$ports = $StartPort..$EndPort

foreach ($port in $ports) {

if ((Test-NetConnection -ComputerName $Server -Port $port).TcpTestSucceeded) {

Write-Host "✅ Port $port is open on $Server"

exit 0

}

}

Write-Host "❌ No open SQL Server port found on $Server in range $StartPort-$EndPort"

exit 1

**✅ 2. Compile to Executable**

To turn the PowerShell script into a .exe file, use **PS2EXE**, a PowerShell module that compiles scripts into executables.

**➤ Install PS2EXE (only once):**

powershell

CopyEdit

Install-Module -Name ps2exe -Scope CurrentUser

**➤ Compile the script:**

powershell

CopyEdit

ps2exe.ps1 -inputFile FindSqlPort.ps1 -outputFile FindSqlPort.exe -noConsole

Now you'll have FindSqlPort.exe that can be called from other apps or the command line like this:

cmd

CopyEdit

FindSqlPort.exe -Server myserver.local -StartPort 1400 -EndPort 1500

**🧪 Output Behavior:**

* If an open port is found, it prints:  
  ✅ Port 1433 is open on myserver.local
* If not, it prints:  
  ❌ No open SQL Server port found on myserver.local in range 1400–1500

You can check the .exe's **exit code** (0 = found, 1 = not found) from other scripts or batch files.

Would you like the compiled .exe to return just the port number (for scripting) instead of full messages?

4o

**You said:**

getting non stop pop ups name resolution of myserver.local failed, need to suppress this

**ChatGPT said:**

To **suppress pop-up errors like "Name resolution of myserver.local failed"** during port scanning in PowerShell, especially from Test-NetConnection, you can:

**✅ Use Try/Catch to Suppress Errors and Avoid Pop-Ups**

Here’s the updated FindSqlPort.ps1 that:

* Accepts parameters.
* Suppresses resolution and connectivity errors.
* Only outputs meaningful info (no pop-ups).

powershell

CopyEdit

param (

[string]$Server,

[int]$StartPort = 1433,

[int]$EndPort = 1445

)

$ErrorActionPreference = "SilentlyContinue" # Suppress non-terminating errors

$ports = $StartPort..$EndPort

foreach ($port in $ports) {

try {

$result = Test-NetConnection -ComputerName $Server -Port $port -WarningAction SilentlyContinue -ErrorAction Stop

if ($result.TcpTestSucceeded) {

Write-Output $port

exit 0

}

} catch {

# Suppress name resolution errors silently

continue

}

}

Write-Host "No open SQL Server port found on $Server in range $StartPort-$EndPort"

exit 1

**🛠 Notes:**

* -WarningAction SilentlyContinue suppresses resolution warnings.
* -ErrorAction Stop makes sure we can catch failures like DNS errors.
* try/catch prevents error pop-ups entirely, even in compiled .exe.

**✅ Then Recompile:**

powershell

CopyEdit

ps2exe.ps1 -inputFile FindSqlPort.ps1 -outputFile FindSqlPort.exe -noConsole

You can now call it like:

cmd

CopyEdit

FindSqlPort.exe -Server myserver.local -StartPort 1433 -EndPort 1500

Let me know if you'd prefer it to just return the port silently (no messages), or include logging.

4o

**You said:**

I need this to be outside th loop

**ChatGPT said:**

Got it — you want to **suppress all name resolution errors once, outside the loop**, rather than handling it on each iteration. Here's the clean solution:

**✅ Updated FindSqlPort.ps1 with Centralized Error Handling**

powershell

CopyEdit

param (

[string]$Server,

[int]$StartPort = 1433,

[int]$EndPort = 1445

)

$ErrorActionPreference = "SilentlyContinue"

# Try resolving the server name once before scanning

try {

[System.Net.Dns]::GetHostEntry($Server) | Out-Null

} catch {

Write-Host "❌ Name resolution failed for '$Server'"

exit 2

}

# Proceed to scan ports only if name resolution succeeded

$ports = $StartPort..$EndPort

foreach ($port in $ports) {

$result = Test-NetConnection -ComputerName $Server -Port $port -WarningAction SilentlyContinue

if ($result.TcpTestSucceeded) {

Write-Output $port

exit 0

}

}

Write-Host "❌ No open SQL Server port found on $Server in range $StartPort-$EndPort"

exit 1

**✅ Key Behavior:**

* DNS resolution is tested **once before the loop**, using [System.Net.Dns]::GetHostEntry().
* If resolution fails, the script exits early with a clean message and exit code 2.
* This avoids repeated Test-NetConnection attempts that would pop up DNS errors.
* The script still returns the first open port if found.

**✅ Recompile as .exe:**

powershell

CopyEdit

ps2exe.ps1 -inputFile FindSqlPort.ps1 -outputFile FindSqlPort.exe -noConsole

Now, calling:

cmd

CopyEdit

FindSqlPort.exe -Server myserver.local -StartPort 1433 -EndPort 1500

...will work cleanly — no DNS popups, clean output, proper exit codes.

Would you like a version that returns just the port as plain text (no message) for scripting?

4o

**You said:**

Remove-Item : Cannot remove item C:\SCRIPTS SQL-SERVER-Main,Current\Port\FindSqlPort.exe: Access to the path 'C:\SCRIPTS SQL-SERVER-Main,Current\Port\FindSqlPort.exe' is denied.

**ChatGPT said:**

The error message you're seeing:

csharp

CopyEdit

Remove-Item : Cannot remove item ...: Access to the path is denied.

means PowerShell cannot delete FindSqlPort.exe because **it’s either currently in use or you lack permissions**.

**✅ Try These Fixes:**

**1. Close any running instances of FindSqlPort.exe**

The .exe cannot be deleted if it's running. Do this:

* Open **Task Manager**
* Look for **FindSqlPort.exe**
* Right-click > **End task**

**2. Ensure PowerShell or your script isn’t using it**

If you’re calling FindSqlPort.exe in your session, **restart PowerShell** before running Remove-Item.

**3. Use Stop-Process (if it’s still stuck)**

In PowerShell:

powershell

CopyEdit

Get-Process FindSqlPort -ErrorAction SilentlyContinue | Stop-Process -Force

Then try:

powershell

CopyEdit

Remove-Item "C:\SCRIPTS SQL-SERVER-Main,Current\Port\FindSqlPort.exe" -Force

**4. Check Permissions**

Make sure your PowerShell session is running **as Administrator**:

* Right-click PowerShell > **Run as Administrator**

**5. Rename then Delete (Workaround)**

Sometimes, renaming first works:

powershell

CopyEdit

Rename-Item "FindSqlPort.exe" "temp.exe"

Remove-Item "temp.exe"