**Task 4 — Windows Firewall: Configuration, Test, and Verification**

**Environment**

* OS: **Windows 11**
* Tools: **Windows Defender Firewall with Advanced Security**, **PowerShell (Admin)**, **Python (http.server)**

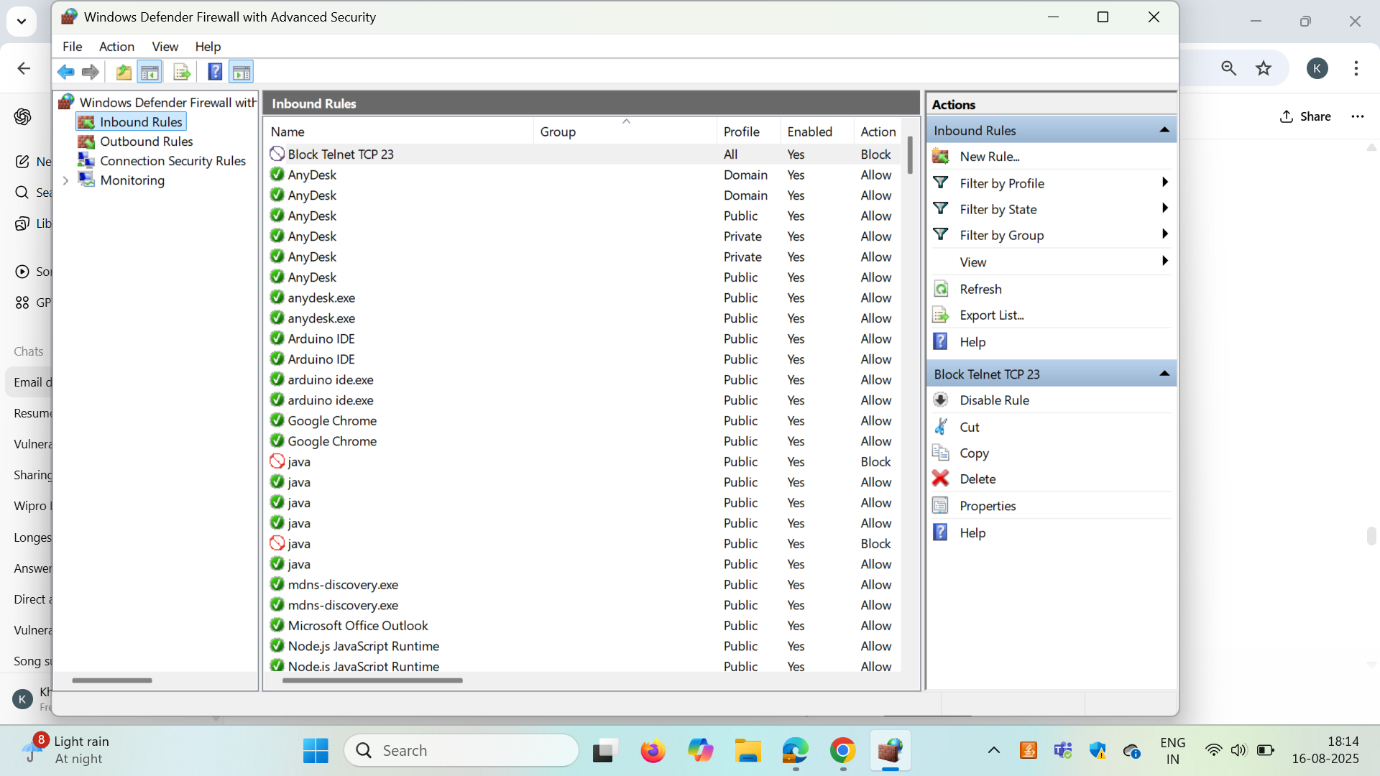
Tip: Open **PowerShell as Administrator** (Start → type *PowerShell* → right-click → *Run as administrator*).

**1) See current firewall state & profiles**

**Command**

Get-NetFirewallProfile | Select Name, Enabled, DefaultInboundAction, DefaultOutboundAction | Format-Table

**Output :**

**Meaning**

* **Enabled=True** → profile is active.
* **DefaultInboundAction=Block** → anything not explicitly allowed is blocked.
* **DefaultOutboundAction=Allow** → apps can go out unless blocked.

**2) Create the “Block Telnet TCP 23” inbound rule (GUI or PowerShell)**

**(A) GUI quick path**

Windows Defender Firewall with Advanced Security → **Inbound Rules** → **New Rule…**  
→ **Port** → TCP → Specific local ports: 23 → **Block the connection** → Apply to Domain/Private/Public → Name: **Block Telnet TCP 23** → Finish.

**(B) PowerShell (one-liner)**

New-NetFirewallRule -DisplayName "Block Telnet TCP 23" -Direction Inbound -Action Block -Protocol TCP -LocalPort 23

**Verify the rule exists**

Get-NetFirewallRule -DisplayName "Block Telnet TCP 23" |

Format-Table DisplayName, Enabled, Direction, Action, Profile

Get-NetFirewallRule -DisplayName "Block Telnet TCP 23" | Get-NetFirewallPortFilter |

Format-Table Protocol, LocalPort

**Meaning**

* Rule is enabled, inbound, action **Block**, on **TCP 23**.

Note: Port 23 (Telnet) is insecure/legacy. Blocking it is a common hardening step.

**3) Prepare a safe test using a temporary local server (port 2323)**

We’ll prove the firewall effect clearly using a port **we control** (2323).  
This avoids running any risky Telnet service on 23.

**Start a listener (in a separate PowerShell window)**

python -m http.server 2323

You should see:

Serving HTTP on :: port 2323 (http://[::]:2323/) ...

**Sanity test (should be open right now)**

Test-NetConnection -ComputerName 127.0.0.1 -Port 2323

**Open state – expected output**

ComputerName : 127.0.0.1

RemoteAddress : 127.0.0.1

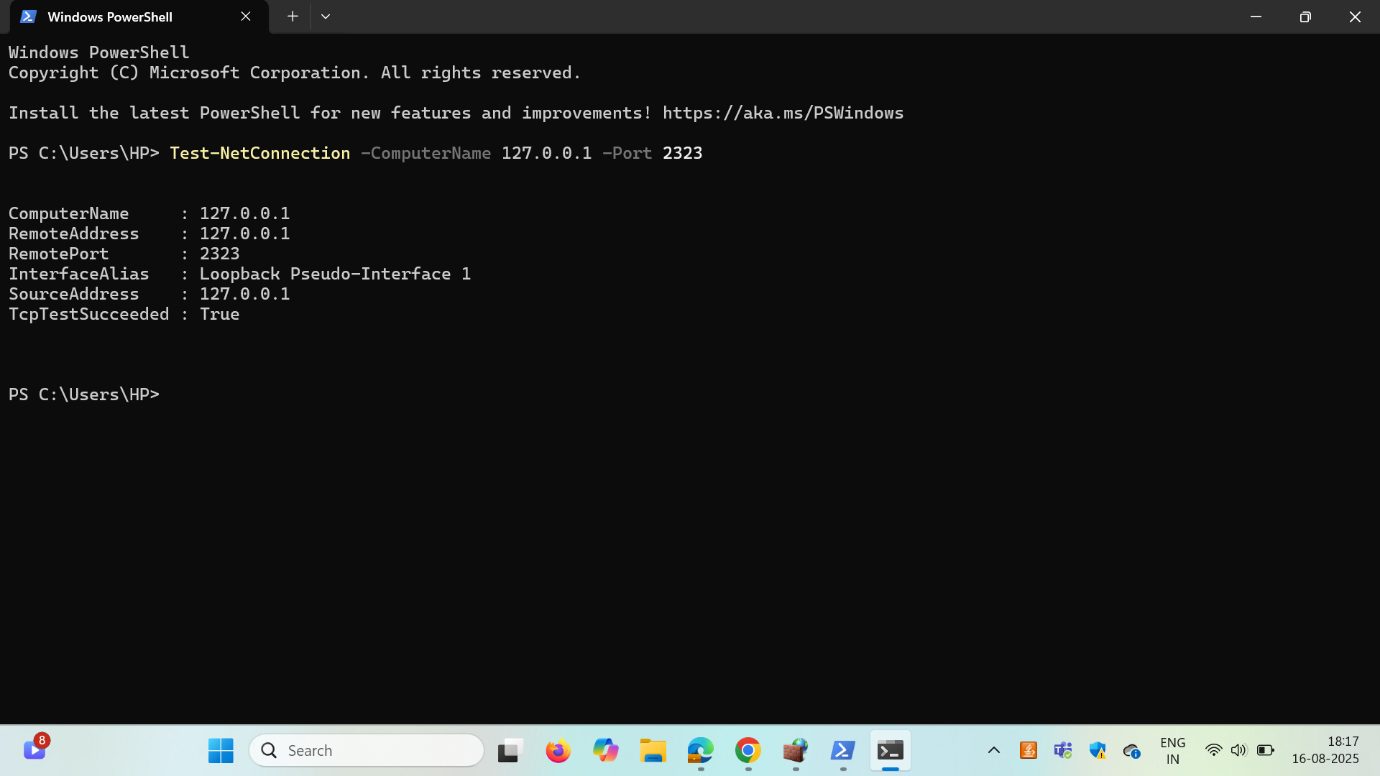
RemotePort : 2323

InterfaceAlias : Loopback Pseudo-Interface 1

SourceAddress : 127.0.0.1

TcpTestSucceeded : True

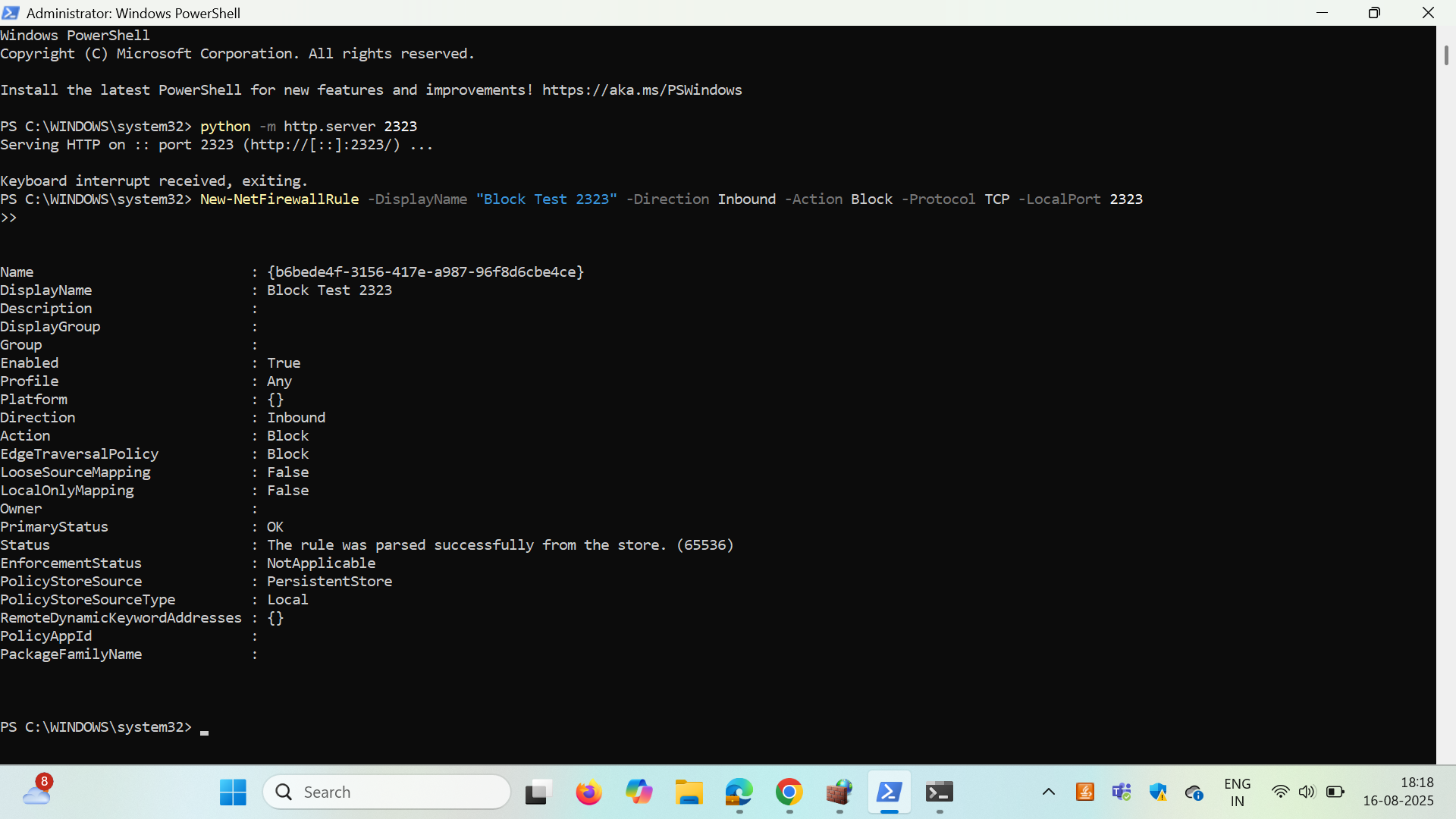
**Meaning (key lines)**

* **RemotePort : 2323** → you tested port 2323
* **TcpTestSucceeded : True** → connection **succeeded** (no block yet)
* 

**4) Create a block rule for our test port (2323) and observe the failure**

**Command (create rule)**

New-NetFirewallRule -DisplayName "Block Test 2323" -Direction Inbound -Action Block -Protocol TCP -LocalPort 2323



**What PowerShell returns (example)**

Name : {…GUID…}

DisplayName : Block Test 2323

Enabled : True

Direction : Inbound

Action : Block

PrimaryStatus : OK

Status : The rule was parsed successfully from the store. (65536)

**Re-test connectivity**

Test-NetConnection -ComputerName 127.0.0.1 -Port 2323

**Blocked state – expected output**

WARNING: TCP connect to (127.0.0.1 : 2323) failed

ComputerName : 127.0.0.1

RemoteAddress : 127.0.0.1

RemotePort : 2323

InterfaceAlias : Loopback Pseudo-Interface 1

SourceAddress : 127.0.0.1

TcpTestSucceeded : False

**Meaning**

* **TcpTestSucceeded : False** → firewall rule is **blocking** inbound traffic to 2323.

**5) Verify rules that target a specific port**

**List all firewall port filters looking at 23**

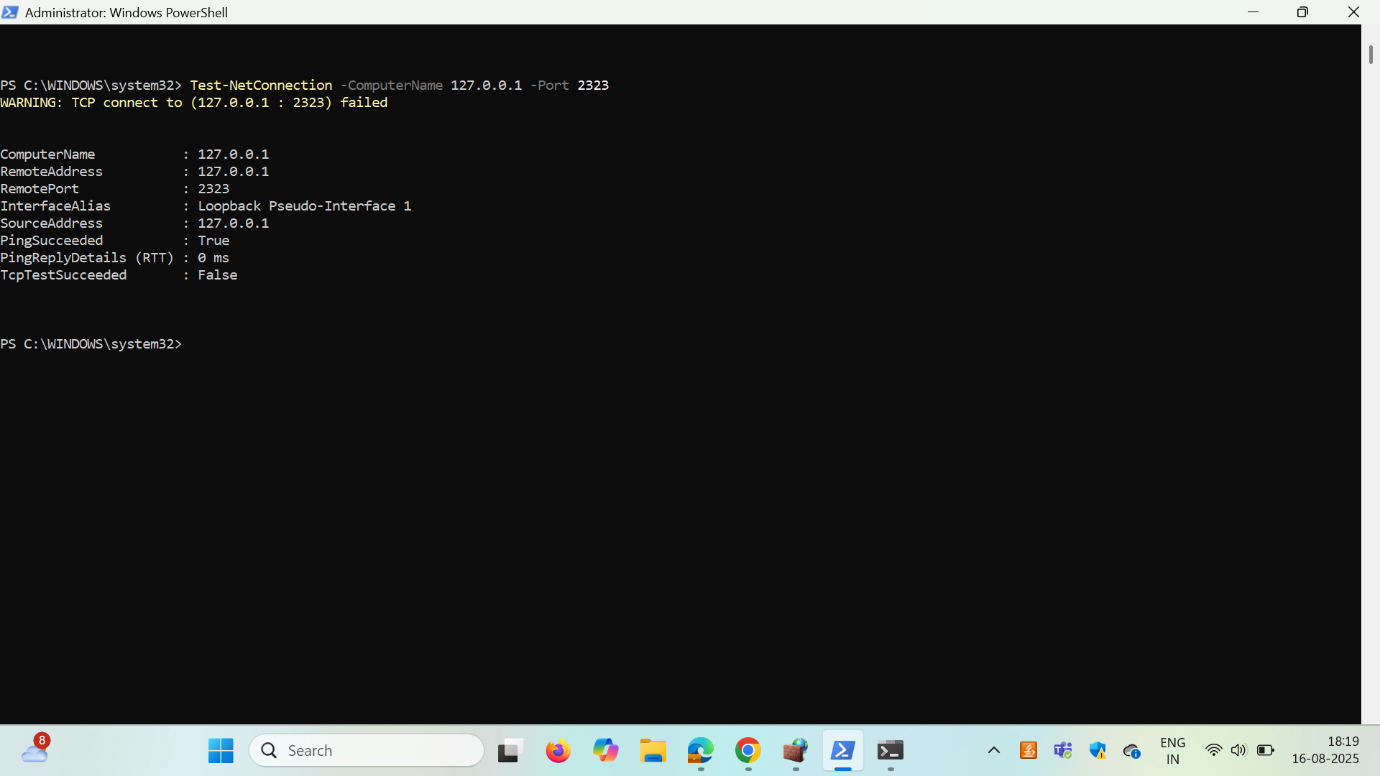
Get-NetFirewallPortFilter | Where-Object LocalPort -eq 23 | Format-Table LocalPort, Protocol, RemotePort

**Find the parent rule names for those filters**

Get-NetFirewallRule |

Where-Object { $\_.DisplayName -like "\*23\*" } |

Format-Table DisplayName, Enabled, Direction, Action



**Meaning**

* Confirms which rules mention **port 23**, their **direction**, and **action**.

**6) Remove (or disable) the test rule and confirm it’s really gone**

**Delete the test rule**

Remove-NetFirewallRule -DisplayName "Block Test 2323"

**Confirm deletion**

Get-NetFirewallRule -DisplayName "Block Test 2323"

* If nothing prints → ✅ the rule is deleted.

**Re-test connectivity (should succeed again)**

Test-NetConnection -ComputerName 127.0.0.1 -Port 2323

**Expected output (success again)**

ComputerName : 127.0.0.1

RemoteAddress : 127.0.0.1

RemotePort : 2323

InterfaceAlias : Loopback Pseudo-Interface 1

SourceAddress : 127.0.0.1

TcpTestSucceeded : True

**Alternative to delete → just disable**

Disable-NetFirewallRule -DisplayName "Block Test 2323"

# Re-enable later:

Enable-NetFirewallRule -DisplayName "Block Test 2323"

**7) Clean-up (keep your system tidy)**

If you only needed Telnet block for the task demo, you can remove it; otherwise keep it (recommended).

# Optional clean up

Remove-NetFirewallRule -DisplayName "Block Telnet TCP 23"

**Export proof of your rules/config (optional for repo)**

Get-NetFirewallRule |

Select DisplayName,Direction,Action,Enabled,Profile |

Export-Csv "$env:USERPROFILE\Desktop\firewall\_rules.csv" -NoTypeInformation

**What the outputs tell you (quick decoder)**

* **“The rule was parsed successfully from the store. (65536)”**  
  → The rule was created & committed to the local firewall policy store.
* **TcpTestSucceeded : True**  
  → TCP handshake completed; firewall didn’t block and a service was listening.
* **TcpTestSucceeded : False + “WARNING: TCP connect … failed”**  
  → No TCP handshake. Usually caused by a firewall block or no listener.  
  In our test, we confirmed a listener existed, so the **firewall** caused the failure.
* **Get-NetFirewallRule -DisplayName "<name>" returns nothing**  
  → The rule is **deleted** (not just disabled).

**Summary (you can paste this into your “Observations/Learnings”)**

* I created **two inbound block rules**: one for **Telnet (TCP 23)** via GUI and one test rule for **TCP 2323** via PowerShell.
* I hosted a temporary local HTTP server on **2323** and verified connectivity with Test-NetConnection.
* With the **block rule active**, PowerShell showed TcpTestSucceeded : False.
* After **removing the rule**, the same test showed TcpTestSucceeded : True.
* I verified rule creation and deletion using Get-NetFirewallRule and Get-NetFirewallPortFilter.
* Key learning: Windows Firewall is **stateful** and enforces **inbound/outbound** policy per profile. Blocking insecure ports (e.g., **23/Telnet**) reduces the attack surface.