**Task – 4**

**Setup and Use a Firewall on Windows/Linux**

Firewall: A firewall is a **security system** (hardware or software) that monitors and controls incoming and outgoing network traffic based on predetermined security rules. Its main purpose is to **allow safe traffic and block harmful or unauthorized traffic** to protect devices and networks from cyber threats.

In Windows, we have both a **graphical user interface (GUI)** and a **command-line tool** to configure the firewall, providing flexibility for users to manage network security either visually or through precise commands.

GUI(Windows Defender Firewall with Advanced Security):

A screenshot of a computer

AI-generated content may be incorrect.

Command-line tool(Powershell):

A screenshot of a computer

AI-generated content may be incorrect.

For this task, the firewall configuration on Windows was performed using the **PowerShell command-line tool**. While Windows also provides a **graphical user interface (GUI)** through *Windows Defender Firewall with Advanced Security*, PowerShell allows precise control over firewall rules. The following documentation outlines the steps taken to configure, test, and remove firewall rules as part of Task 4.

To begin the task, we first reviewed the existing firewall rules to understand the current configuration before making any changes. This ensures clarity on what rules already exist and avoids conflicts when adding new ones.

On Windows, this was accomplished using PowerShell with the following command:

* **netsh advfirewall firewall show rule name=all**

This command displays a complete list of all inbound and outbound firewall rules currently configured on the system, including their names, directions, actions, protocols, and ports.

A screenshot of the command output was captured to document the initial firewall state before proceeding to create new rules.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

**Blocking Telnet (Port 23)**

Before blocking Telnet, we first enabled the Telnet client using the **Windows Firewall GUI**. By default, Telnet is disabled in Windows, so enabling it ensured we could test the block rule properly.

To enable Telnet via GUI:

1. Open **Control Panel**.
2. Navigate to **Programs → Turn Windows features on or off**.
3. Check **Telnet Client** and click **OK**.

A screenshot of a computer program

AI-generated content may be incorrect.

Once Telnet was enabled, we added a firewall rule to block inbound traffic on port 23 using PowerShell:

* **netsh advfirewall firewall add rule name="Block Telnet" dir=in action=block protocol=TCP localport=23**

This command created a rule named “Block Telnet” that denies all inbound TCP traffic on port 23.

We tested the rule by attempting to connect to Telnet locally:

* **telnet localhost 23**

The connection failed, confirming that the block rule was successfully applied.

Finally, we removed the rule to restore the firewall state:

* **netsh advfirewall firewall delete rule name="Block Telnet"**

Screenshots were taken before and after adding/deleting the rule to document the change

A black screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A black screen with white text

AI-generated content may be incorrect.

**Blocking a Python Web Server (Port 8080)**

To further demonstrate firewall rule configuration, we set up a simple web server on Windows using Python and then applied a block rule for its port.

**Start the Web Server**

We opened PowerShell and ran:

* **python -m http.server 8080**

This started a basic HTTP server listening on port 8080. We confirmed it was running by opening a browser and visiting:

* http://localhost:8080

The page loaded successfully, verifying the server was active.

**Add the Block Rule**

To block inbound traffic on port 8080, we ran:

* **netsh advfirewall firewall add rule name="Block WebServer" dir=in action=block protocol=TCP localport=8080**

However, testing showed that the block was not effective locally due to Windows Firewall ignoring local loopback traffic.

**Add Additional Rules**

To attempt a complete block, we added two more rules:

* **netsh advfirewall firewall add rule name="Block WebServer In" dir=in action=block protocol=TCP localport=8080**
* **netsh advfirewall firewall add rule name="Block WebServer Out" dir=out action=block protocol=TCP remoteport=8080**

These rules aimed to block both inbound and outbound traffic for the web server port.

**Step 4 — Test the Rules**

Despite adding these rules, the block was still not visible when tested locally. This is due to Windows Firewall’s handling of local loopback traffic — the block would be effective only for external connections from another device on the network.

**Step 5 — Delete the Block Rules**

After testing, we deleted all three rules to restore normal traffic:

* **netsh advfirewall firewall delete rule name="Block WebServer"**
* **netsh advfirewall firewall delete rule name="Block WebServer In"**
* **netsh advfirewall firewall delete rule name="Block WebServer Out"**

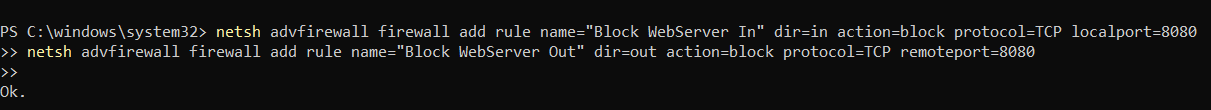
Screenshots of the firewall rules added and deleted were captured for documentation purposes:

A computer screen with white text

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.A screen shot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

**Deliverables & Summary**

For Task 4, we configured and tested firewall rules on Windows using **PowerShell** as the command-line configuration tool. Screenshots of each step were captured for documentation.

**Deliverables:**

1. **List of existing firewall rules** before any changes.
2. **Block rule for Telnet (Port 23)** — added and tested successfully.
3. **Three block rules for Python web server (Port 8080)** — added to attempt blocking, with explanation why they did not work locally.
4. **Deletion of all block rules** to restore original state.
5. Screenshots showing commands, rule additions, and rule deletions.

**Summary of Findings:**

* Windows Firewall can be configured via both **GUI** and **command-line tools** (netsh advfirewall).
* Blocking inbound TCP ports (e.g., Telnet port 23) works successfully and can be tested locally.
* Blocking local services (e.g., Python HTTP server on port 8080) requires extra rules because Windows Firewall does not filter **local loopback traffic** by default.
* Multiple rules can be added to block both inbound and outbound traffic, but local testing may still bypass the block.
* Deleting firewall rules restores normal traffic flow.

**Conclusion:**

Firewalls are essential for network security as they allow control over inbound and outbound traffic. Through this task, we learned how to list, add, test, and delete firewall rules in Windows, and understood limitations related to local traffic filtering.