PowerShell Bot Explanation

Overview

The NetworkDeviceBot class is a PowerShell-based automation tool designed to scan, access, and interact with devices on a local network. It provides functionalities such as:

- Device discovery and connectivity checking
- Network scanning
- Remote access via PowerShell, SSH, SNMP, and WMI
- Port scanning
- Shared folder access
- Credential management for authenticated operations

The bot aims to automate network monitoring and administration while maintaining security and error handling.

1. Structure and Functionality

1.1 Properties

The class maintains three key properties:

Property	Description
IPRange	Defines the subnet to scan (default: 192.168.1.x).
MaxPorts	The number of ports to scan when checking for open services.
Credentials	Stores authentication credentials for accessing remote devices.

1.2 Methods and Their Functions

1.2.1 Device Connectivity Check

```
[PSCustomObject] CheckDeviceConnectivity([string]$ipAddress)
```

- Test connection (ICMP ping) is used to check if a device is reachable.
- Test-NetConnection is used to verify whether TCP port 80 is open.
- Returns structured data with:
    - IP Address
    - Reachability Status
    - Response Time
    - Port 80 Status
    - Error Handling

1.2.2 Network Scanning

```
[Array] ScanNetwork()
```

- Iterates through 192.168.1.1 to 192.168.1.254 (or another defined subnet).
- Uses Test-Connection to check active devices.
- Returns a list of online devices.

1.2.3 Access Shared Folders

```
[PSCustomObject] AccessSharedFolder([string]$ipAddress, [string]$shareName)
```

- Uses New-PSDrive to map network shares as local drives.
- Useful for managing shared files across multiple devices.

1.2.4 Execute Remote PowerShell Commands

```
[PSCustomObject] ExecuteRemoteCommand([string]$ipAddress, [ScriptBlock]$scriptBlock)
```

- Uses Invoke-Command to run commands remotely via PowerShell Remoting (WinRM).
- Requires administrator permissions on the remote machine.

1.2.5 Scan Ports

```
[Array] ScanPorts([string]$ipAddress)
```

- Iterates through 1 to MaxPorts (default: 1024).
- Uses Test-NetConnection to identify open ports.
- Returns a list of open ports.

1.2.6 Get SNMP Information

```
[PSCustomObject] GetSNMPInfo([string]$ipAddress)
```

- Uses SNMP queries to fetch system information (e.g., uptime, device type).
- Requires SNMP to be enabled on the target device.

1.2.7 Execute SSH Command

```
[PSCustomObject] ExecuteSSHCommand([string]$ipAddress, [string]$command)
```

- Uses PowerShell's built-in SSH client to run commands on Linux-based systems or network devices.
- Useful for managing routers, switches, and Linux servers.

1.2.8 Get WMI Information

```
[PSCustomObject] GetWMIInfo([string]$ipAddress)
```

- Uses WMI (Get-WmiObject) to gather system details like CPU, RAM, and OS information.
- Requires administrator privileges on Windows targets.

2. Example Usage

The Bot is used as follows:

```
# Create a new instance of the bot
$bot = [NetworkDeviceBot]::new()

# Scan the network for devices
$devices = $bot.ScanNetwork()

# Check connectivity for a specific device
$deviceStatus = $bot.CheckDeviceConnectivity("192.168.1.10")

# Access a shared folder
$shareResult = $bot.AccessSharedFolder("192.168.1.10", "SharedFolder")

# Execute a remote PowerShell command
$remoteResult = $bot.ExecuteRemoteCommand("192.168.1.10", { Get-Process })

# Scan ports
$openPorts = $bot.ScanPorts("192.168.1.10")

# Get SNMP information
$snmpInfo = $bot.GetSNMPInfo("192.168.1.10")

# Execute SSH command
$sshResult = $bot.ExecuteSSHCommand("192.168.1.10", "ls -l")

# Get WMI information
$wmiInfo = $bot.GetWMIInfo("192.168.1.10")
```

3. Security Features

- Centralized authentication: Credentials property manages authentication securely.

- Exception handling: Each method includes error handling to prevent script crashes.
- Secure communication: Uses secure protocols such as WinRM, SSH, and SNMP.
- Minimal privilege principle: Ensures the least privileged access to remote systems.

4. Potential Enhancements

- Multi-threading: Parallel execution for faster network scanning.
- Logging and reporting: Store scan results in a file for analysis.
- Customization options: Allow dynamic port range selection.
- GUI interface: A PowerShell GUI for easier interaction.