PowerShell Security Report Script

This script performs a comprehensive security analysis of your Windows system by running 22 crucial checks. It's designed to help system administrators and security professionals ensure their Windows environments are secured according to best practices and identify potential vulnerabilities.

Features

The script checks the following system parameters and configurations:

1. **Windows Firewall Status**: Verifies if the firewall is active to block unauthorized access.
2. **User Account Control (UAC) Status**: Confirms that UAC is enabled to prevent unauthorized changes.
3. **Windows Defender Status**: Checks if Windows Defender is active and updated.
4. **BitLocker Status**: Verifies if BitLocker is enabled for data protection.
5. **Operating System Updates**: Ensures that the OS is current with updates.
6. **Password Policy**: Evaluates the robustness of the system's password policy.
7. **Failed Login Attempts**: Checks for multiple failed login attempts.
8. **Admin Users**: Lists all users with administrative privileges.
9. **Guest Account Status**: Confirms the guest account is disabled when not in use.
10. **Network Shares**: Reviews all network shares and their permissions.
11. **AutoRun Status**: Checks if AutoRun is disabled.
12. **Remote Desktop Status**: Verifies the status of Remote Desktop and its configuration.
13. **Antivirus Software Status**: Checks for third-party antivirus software and its status.
14. **System Restore Point**: Checks for the latest system restore point.
15. **Security Services Status**: Confirms critical security services are running.
16. **Open Ports**: Lists open network ports and their services.
17. **Windows Event Logs**: Reviews system and security logs for significant incidents.
18. **Scheduled Tasks Analysis**: Reviews scheduled tasks with high-level privileges.
19. **Secure Boot Status**: Verifies if Secure Boot is enabled.
20. **Active Network Connections**: Reviews all active network connections.
21. **Startup Applications**: List all startup applications.
22. **Scheduled Tasks**: List any scheduled tasks that have run at least once.

Requirements

- Windows PowerShell 5.1 or higher
- Appropriate administrative privileges to execute system checks

Usage

1. Download the script or copy it into a new **.ps1** file on your system.
2. Open PowerShell with administrative privileges.
3. Navigate to the directory containing the script.
4. Run the script by entering **.\script_name.ps1** (replace **script_name.ps1** with the actual script filename).

The script will execute and provide output for each of the 20 security checks. Depending on system configuration, some checks may require confirmation or additional input.

Output

The script generates its findings in real-time, writing the output to the console or a specified output file if configured. Review the results to understand your system's security posture and identify any areas needing improvement.

Caution

This script performs read-only checks and does not modify system settings. However, it's advisable to review the script thoroughly before running it in a production environment. Always follow your organization's policies and procedures when making changes to system configurations.

Disclaimer

Use this script at your own risk. The creator(s) cannot be held responsible for any issues that arise from its use.

Running the Script with Elevated Permissions

This script requires elevated permissions to perform comprehensive system checks. To run the script with administrative privileges, follow these steps:

1. **Launch PowerShell as Administrator**:
    - Right-click the Start button or press **Win + X**.
    - Choose “Windows PowerShell (Admin)” or “Command Prompt (Admin)” if PowerShell is not available.
    - Confirm any User Account Control (UAC) prompts if necessary.
2. **Navigate to the Script's Directory**:
    - Use the **cd** command to navigate to the directory where your script is located.
3. **Run the Script**:
    - Execute the script by typing **.\script_name.ps1** (replace **script_name.ps1** with the actual script filename) and press Enter.

Bypassing the Execution Policy

By default, PowerShell restricts script execution to protect against unauthorized scripts. If you encounter a code signing error, it's likely due to the execution policy. To bypass this for the script, follow these steps:

1. **Understand the Implications**:
    - Bypassing the execution policy can expose your system to risks. Understand the script source and content before proceeding. Never bypass the execution policy for scripts from untrusted sources.
2. **Set the Execution Policy for the Session**:

- Instead of changing the execution policy for your system, you can bypass it for a single session. This method is safer and does not require a permanent change to your system settings.
- After launching PowerShell with administrative privileges, type the following command and press Enter:

sqlCopy code

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

- This command only bypasses the execution policy for the current PowerShell session.

3. **Run the Script**:
    - After setting the execution policy, navigate to the script's directory.
    - Execute the script by typing **.\script_name.ps1** (replace **script_name.ps1** with the actual script filename) and press Enter.

Remember, these instructions temporarily bypass the script execution policy for your current session but do not change your system's configuration. Always ensure that the scripts you run are from trusted sources and have been reviewed for security and functionality.

This information ensures users can successfully run your security script, understanding the necessary permissions and how to safely bypass execution policy restrictions.

Running a PowerShell script across multiple PCs on a network from a Domain Controller requires careful execution. You have to ensure that you do not compromise security while also achieving the desired task on all client machines. Here are the steps to achieve this, along with appropriate warnings:

Pre-requisites:

- Administrative credentials on the Domain Controller and permissions to execute scripts on all target PCs.
- Ensure all target PCs are domain-joined and connected to the network.
- A shared folder that is accessible by the administrator and the target PCs.

Steps:

1. **Place the Script in a Shared Folder**:
    - Save the PowerShell script (**script_name.ps1**) in a shared folder on the network which is accessible by the target PCs.
2. **Launch PowerShell as Administrator on the Domain Controller**:
    - On the Domain Controller, right-click the Start button or press **Win + X**.
    - Choose "Windows PowerShell (Admin)".
    - Confirm any User Account Control (UAC) prompts if necessary.
3. **Set Execution Policy**:
    - Run the following command to temporarily change the execution policy for the current PowerShell session:

PowershellCopy code

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Bypass
```

This command only affects the current session and does not make permanent changes to the system's execution policy settings.

⚠ **Warning**: This allows the current PowerShell session to run scripts, potentially including malicious ones. Only proceed if the script source is trusted.

4. **Create a List of Target PCs**:
    - Create a text file (**targets.txt**) containing the hostnames or IP addresses of all target PCs, each on a new line.
    - You can use the **QueryADforPCs powershell** script to generate **targets.txt,** reference it's readme.
5. **Execute the Script Remotely**:
    - Use the following script to execute **script_name.ps1** on each PC listed in **targets.txt**:

powershellCopy code

```
$Computers = Get-Content "path\to\targets.txt" foreach ($Computer in $Computers) { Invoke-Command -ComputerName $Computer -FilePath "\\path\to\shared\folder\script_name.ps1" -ErrorAction Stop }
```

Make sure to replace **"path\to\targets.txt"** and **"\\path\to\shared\folder\script_name.ps1"** with the actual paths to your target list and PowerShell script, respectively.

⚠ **Warning**: The **Invoke-Command** cmdlet runs commands on a local or remote computer and returns all output from the commands, including errors. Using **-ErrorAction Stop** will stop the script execution upon encountering any error.

6. **Review the Output**:
    - Check the output of the script for each machine to verify the script ran successfully and to review any data or messages it returned.
7. **Reset Execution Policy (Optional)**:
    - After the script execution, you can reset the execution policy to the default setting by closing the PowerShell window or typing:

powershellCopy code

```
Set-ExecutionPolicy -Scope Process -ExecutionPolicy Restricted
```

Then, confirm the action.

⚠ **Security Consideration**: Ensure that the script does not contain sensitive information, as it resides in a shared folder. After the operation, it's advisable to remove the script from the shared location or restrict its access.

By following these steps, you can execute your PowerShell script on multiple PCs within a network from a Domain Controller. Always exercise caution with script content, execution policies, and network commands to maintain system integrity and security.