**CIS BENCHMARKS**

CIS (Centre for Internet Security) Benchmarks are a set of best practices and guidelines developed to help organizations secure their systems and data. They are globally recognized as standards for securing IT systems and data against cyber threats.

**AN OVERVIEW**

1. PURPOSE

- To provide a comprehensive set of security configurations that help organizations improve their cybersecurity posture.

- To ensure that IT systems are configured securely out of the box, minimizing vulnerabilities.

2. DEVELOPMENT

- Created and maintained by cybersecurity experts, IT professionals, and the broader security community.

- Regularly updated to address new threats and technological changes.

3. SCOPE

- Covers a wide range of technologies including operating systems, cloud infrastructure, network devices, applications, and more.

- Includes benchmarks for systems like Windows, Linux, macOS, AWS, Azure, GCP, and various database and web server platforms.

**KEY FEATURES**

1. CONFIGURATION GUIDELINES

- Provides detailed configuration settings and practices to secure systems.

- Includes recommendations for secure installation, configuration, and management.

2. SECURITY CONTROLS

- Aligns with widely recognized frameworks and standards, such as NIST, ISO/IEC, and others.

- Helps in achieving compliance with various regulatory requirements like GDPR, HIPAA, and PCI-DSS.

3. SCORING AND ASSESMENT

- Offers tools like CIS-CAT (CIS Configuration Assessment Tool) for assessing compliance with the benchmarks.

- Provides scoring mechanisms to evaluate the security posture of systems.

4. COMMUNITY AND SUPPORT

- Supported by a large community of cybersecurity professionals.

- Regular updates and community-driven improvements ensure the benchmarks remain relevant and effective.

**ADVANTAGES**

1. Enhanced Security

- Reduces the risk of cyberattacks by addressing common vulnerabilities and misconfigurations.

- Provides a solid foundation for a secure IT environment.

2. Standardization

- Promotes consistency in security configurations across an organization’s infrastructure.

- Facilitates easier management and auditing of security practices.

3. Compliance

- Helps organizations meet regulatory and compliance requirements efficiently.

- Provides a structured approach to achieving and maintaining compliance.

4. Cost-Effective

- Saves time and resources by providing pre-defined, expert-validated security configurations.

- Reduces the need for extensive in-house security expertise.

**IMPLEMENTATION**

1. Assessment

- Evaluate current system configurations against CIS Benchmarks using tools like CIS-CAT.

2. Remediation

- Implement the recommended configurations to address identified gaps and vulnerabilities.

3. Continuous Monitoring

- Regularly assess and update system configurations to ensure ongoing compliance with the benchmarks.

- Stay informed about updates to the CIS Benchmarks and apply changes as necessary.

Using CIS Benchmarks as part of your data security strategy can significantly enhance your organization's ability to protect sensitive data and defend against cyber threats.

CIS Benchmarks themselves aren't code, but rather configuration guides for different operating systems.

Step-by-step process:

1. Choose your Operating System(s):

* CIS Benchmarks cover a wide range of OSes including Windows, Linux distributions (CentOS, Ubuntu, etc.), macOS, and more.

2. Download CIS Benchmarks:

* Download the relevant benchmark for your chosen OS. These are typically available in PDF format.

3. Analyze Benchmark Controls:

* Each CIS Benchmark is divided into sections with specific controls. These controls detail secure configuration options for various system aspects like user accounts, password policies, file permissions, etc.
* Carefully analyze the controls relevant to your needs.

4. Scripting Language Selection:

* Popular options include:
  + PowerShell (Windows)
  + Bash (Linux/macOS)
  + Python (cross-platform)

I have chosen Python

5. Script Development:

* General outline:
  + Import Libraries (if needed): Depending on your language, you might need to import libraries for system interaction (e.g., WMI for Windows, sys for Linux/macOS)
  + Define Functions: Create functions for each control you want to check/enforce.
    - Each function should use system commands or configuration files to access and potentially modify settings based on the CIS Benchmark recommendations.
  + Read Benchmark Data: You can parse the downloaded CIS Benchmark PDF (more challenging) or create a separate configuration file listing the controls and desired states.
  + Loop Through Controls: Use a loop to iterate through each control in your data source.
  + Call Control Functions: Within the loop, call the corresponding function you defined for each control to check and potentially fix the configuration.
  + Reporting: Add functionalities to log results (compliant/non-compliant) and potentially provide remediation steps.

6. Testing on Different OSes:

* Once your script is functional for one OS, adapt it for the others. As commands and configuration files might differ between OSes. You'll need to use OS-specific script.

7. Script Refinement:

* Test your script thoroughly on different systems of your target OSes.
* Refine the script based on test results, fixing errors and handling edge cases.
* Add error handling to gracefully manage situations where configuration checks or modifications fail.

**CIS BENCHMARK TESTING ON WINDOWS 11**

LANGUAGE – PYTHON

1. Choose the type of benchmark to use

|  |  |  |
| --- | --- | --- |
|  | Stand Alone | Enterprise |
| Target Audience | Individual users or small organization | Larger organization with multiple systems in single domain |
| Features | Secure single windows 11 machine | Has adv setting to leverage group policy, active directory, and other enterprise management tools |
| Use Cases | PC, small biz without domain | Corporate envt with an it dept |

1. Review the Benchmark PDF
2. Choose the Benchmarks you want to test and note the control IDs
3. Write python script

BENCHMARK 1

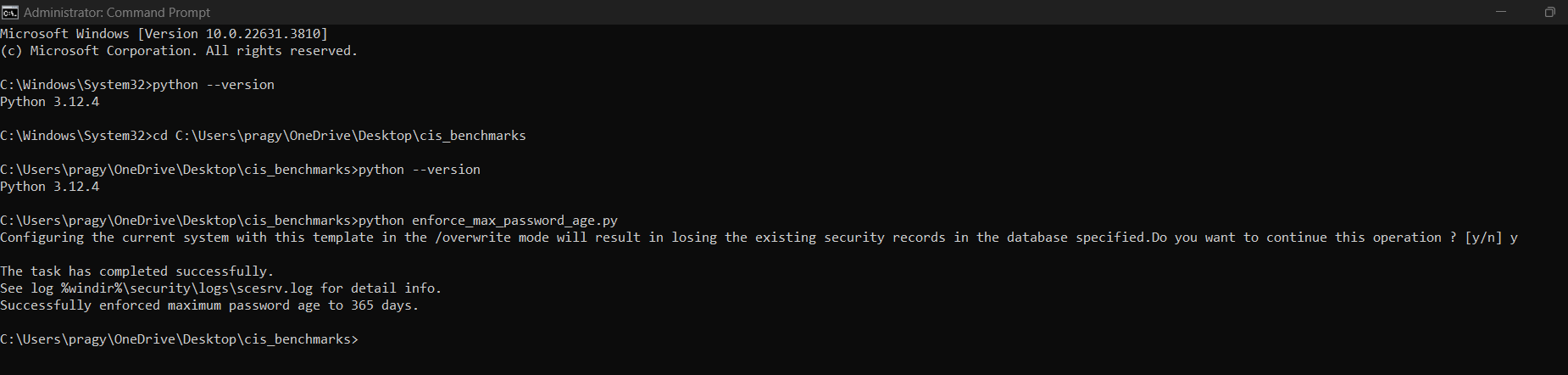
ENSURE ‘ENFORCE PASSWORD HISTORY’ IS SET TO 24 OR MORE

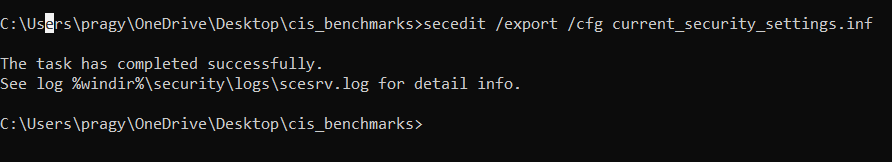
* 1. Write the python code
  2. Run the script as administrator from command prompt:

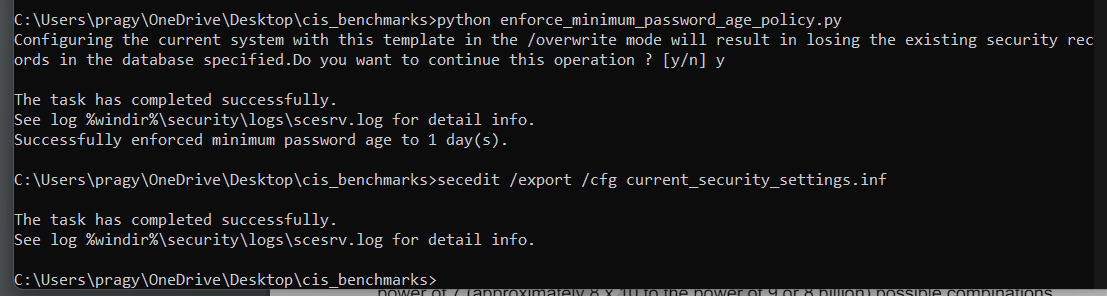
python enforce\_max\_password\_age.py

* 1. Check results using same command prompt and secedit tool:

secedit /export /cfg current\_security\_settings.inf







**ENFORCING CIS BENCHMARKS**

this is a script I wrote to check the maximum password age, it gives the output by printing the admin configured settings. now for the above code of account lockout duration I want the similar output, change the code accordingly.

ACCOUNT POLICIES

* 1. Password Policies (page 32)
     1. Ensure 'Enforce password history' is set to '24 or more password(s)'
     2. Ensure 'Maximum password age' is set to '365 or fewer days, but not 0'
     3. Ensure 'Minimum password age' is set to '1 or more day(s)' (Automated)
  2. Account Lockout Policy (page 50)
     1. Ensure 'Account lockout duration' is set to '15 or more minute(s)’
     2. Ensure 'Account lockout threshold' is set to '5 or fewer invalid logon attempt(s), but not 0'

USER RIGHTS ASSIGNMENT

2.1 user rights (page 60)

2.1.1 Ensure 'Act as part of the operating system' is set to 'No One'

2.1.2 Ensure 'Allow log on locally' is set to 'Administrators, Users'

SECURITY OPTIONS

3.1 Accounts (page 140)

3.1.1 Ensure 'Accounts: Guest account status' is set to 'Disabled

3.1.2 Configure 'Accounts: Rename administrator account'

3.2 Audit

3.2.1 Ensure 'Audit: Shut down system immediately if unable to log security audits' is set to 'Disabled'

3.3 DCOM  
3.4 Devices

3.4.1 Ensure 'Devices: Prevent users from installing printer drivers' is set to 'Enabled'

3.5 Domain Controller

3.6 Domain member

3.7 Interactive logon

3.7.1 Ensure 'Interactive logon: Do not require CTRL+ALT+DEL' is set to 'Disabled'

3.7.2 Ensure 'Interactive logon: Don't display last signed-in' is set to 'Enabled

3.8 Microsoft Network Client

3.8.1 Ensure 'Microsoft network client: Digitally sign communications (always)' is set to 'Enabled

3.9 Microsoft Network Server

3.9.2 Ensure 'Microsoft network server: Digitally sign communications (always)' is set to 'Enabled'

3.10 Network Access

3.10.1 Ensure 'Network access: Allow anonymous SID/Name translation' is set to 'Disabled'

3.11 Network Security

3.11.1 Ensure 'Network security: Allow Local System to use computer identity for NTLM' is set to 'Enabled'

3.11.2 Ensure 'Network security: Allow LocalSystem NULL session fallback' is set to 'Disabled'

3.12 Recovery Console

3.13 Shutdown

3.14 System Cryptography

3.14.1 Ensure 'System cryptography: Force strong key protection for user keys stored on the computer' is set to 'User is prompted when the key is first used' or higher

3.15 System Objects

3.15.1 Ensure 'System objects: Require case insensitivity for non-Windows subsystems' is set to 'Enabled'

3.16 System Settings

3.17 User Account Control

3.17.1 Ensure 'User Account Control: Detect application installations and prompt for elevation' is set to 'Enabled'

CHECKING PATH

Involves checking for updates, security patches or latest versions of installed software to ensure that system is up to date and secure.

The process usually entails:

1. Identifying Installed Software: Listing all the installed applications and their current versions.
2. Comparing Versions: Comparing the installed versions against the latest available versions from the software vendors.
3. Verifying Updates: Checking if there are any missing updates or patches for the installed software.
4. Installing Updates: Applying the latest patches or updates to ensure the software is secure and has the latest features.

CHECK\_PATH.PY SCRIPT

1. **Logging Setup**:
   * Configures logging to capture the script’s actions and any errors.
2. **WMIC Command**:
   * Uses the wmic qfe list brief command to query the list of installed patches. qfe stands for Quick Fix Engineering, which represents updates or hotfixes installed on the system.
   * capture\_output=True captures the output of the command.
   * text=True ensures the output is treated as text.
   * check=True raises an error if the command fails.
3. **Processing Output**:
   * Splits the command output into lines.
   * Prints and logs the header (first line).
   * Prints and logs each subsequent line, which represents an installed patch.
4. **Error Handling**:
   * Catches and logs any errors that occur during the execution of the wmic give me an explanation as to what output this script would give

The provided Python script uses the wmic tool to query and list installed patches (updates) on a Windows system. When you run the script, it will execute the wmic qfe list brief command, which retrieves information about the installed Quick Fix Engineering (QFE) updates, commonly known as hotfixes or patches.

CIS BENCHMARKS FOR BLUETOOTH

1. Disable Bluetooth when not required

- Rationale: Disabling Bluetooth reduces the attack surface and prevents unauthorized devices from connecting to the system.

- Audit: Verify that Bluetooth is turned off.

2. Disable Bluetooth Discovery Mode

- Rationale: Discovery mode allows other devices to see and attempt to connect to the system. Disabling it reduces the risk of unauthorized access.

- Audit: Verify that the system is not discoverable.

3. Enable Bluetooth Security Settings

- Rationale: Ensuring that Bluetooth security settings are enabled helps prevent unauthorized access and data transfer.

- Audit: Verify that Bluetooth security settings are enabled.

4. Remove Paired Bluetooth Devices That Are No Longer Needed

- Rationale: Removing unnecessary paired devices reduces the risk of unauthorized access.

- Audit: Verify that only necessary Bluetooth devices are paired.

5. Ensure Bluetooth PAN (Personal Area Network) is Disabled

- Rationale: Disabling Bluetooth PAN reduces the risk of unauthorized network access via Bluetooth.

- Audit: Verify that Bluetooth PAN is disabled.

**Audit:**

To audit the Bluetooth settings, you can manually check the Bluetooth settings in the Windows Settings or use PowerShell to retrieve the current status:

**Remediation:**

To ensure compliance with the CIS benchmarks, you can implement scripts and group policies to enforce the required settings.

These steps help ensure that Bluetooth settings on a Windows 11 standalone system are configured securely, reducing the risk of unauthorized access and enhancing overall system security. For detailed and up-to-date recommendations, refer to the official CIS Benchmark documentation for Windows 11.

SYSTEM AUDIT SCRIPT OVERVIEW

I am writing a Python script to perform a system audit on a Windows machine, gathering information such as admin users, user privileges, installed patches, current logged-in users, and installed software. The script uses subprocess calls to run system commands and queries the Windows registry to collect data, logging all activities and errors for thorough documentation. By retrieving and analyzing this information, the script helps ensure system security and compliance with configuration standards.

This same task is done for:

* + - * Windows 11
      * Windows 10
      * Windows 8
      * Windows 7