CIS Center for Internet Security®

CIS Benchmarks™

CIS Microsoft DotNet Framework 4.0 STIG Benchmark

v1.0.0 - 09-03-2025

Terms of Use

Please see the below link for our current terms of use:

https://www.cisecurity.org/cis-securesuite/cis-securesuite-membership-terms-of-use/

For information on referencing and/or citing CIS Benchmarks in 3rd party documentation (including using portions of Benchmark Recommendations) please contact CIS Legal (legalnotices@cisecurity.org) and request guidance on copyright usage.

NOTE: It is ***NEVER*** acceptable to host a CIS Benchmark in ***ANY*** format (PDF, etc.) on a 3rd party (non-CIS owned) site.

Table of Contents

Overview

Target Technology Details

Microsoft DotNet Framework 4.0 Secure Technical Implementation Guide (STIG)

Version: 2 Release: 6 Benchmark

Date: 02 Apr 2025

Intended Audience

This benchmark is intended for system and application administrators, security specialists, auditors, help desk, and platform deployment personnel who plan to develop, deploy, assess, or secure solutions that incorporate Microsoft DotNet Framework 4.0 and are looking to comply with the STIG guidance

Recommendation Definitions

The following defines the various components included in a CIS recommendation as applicable. If any of the components are not applicable it will be noted, or the component will not be included in the recommendation.

Title

Concise description for the recommendation's intended configuration.

Assessment Status

An assessment status is included for every recommendation. The assessment status indicates whether the given recommendation can be automated or requires manual steps to implement. Both statuses are equally important and are determined and supported as defined below:

Automated

Represents recommendations for which assessment of a technical control can be fully automated and validated to a pass/fail state. Recommendations will include the necessary information to implement automation.

Manual

Represents recommendations for which assessment of a technical control cannot be fully automated and requires all or some manual steps to validate that the configured state is set as expected. The expected state can vary depending on the environment.

Profile

A collection of recommendations (or rules) for securing a technology or a supporting platform. STIG Benchmark profiles are used to identify which Vulnerability Severity Category Code (CAT) each rule is associated with.

Description

The Rule Title from the STIG.

Rationale Statement

Detailed reasoning for the recommendation to provide the user a clear and concise understanding on the importance of the recommendation.

Audit Procedure

Systematic instructions for determining if the target system complies with the recommendation.

Remediation Procedure

Systematic instructions for applying recommendations to the target system to bring it into compliance according to the recommendation.

Additional Information

References from the STIG Rule if applicable.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **SEVERITY: CAT I**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be high severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

- **SEVERITY: CAT II**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be medium severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

- **SEVERITY: CAT III**

  Items in this profile exhibit one or more of the following characteristics:

  - are considered to be low severity
  - are intended for environments or use cases where following STIG based security guidance is paramount.
  - acts as defense in depth measure.
  - may negatively inhibit the utility or performance of the technology.

  This profile is intended for servers and workstations.

Acknowledgements

The Recommendations in this Benchmark are a representation of the Rules in the unclassified DISA STIG for Microsoft DotNet Framework 4.0

Recommendations

1 STIG RULES

Microsoft DotNet Framework 4.0

Microsoft DotNet Framework 4.0 Secure Technical Implementation Guide (STIG)

Version: 2 Release: 6 Benchmark

Date: 02 Apr 2025

CLASSIFICATION unclassified

1.1 APPNET0031 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Digital signatures assigned to strongly named assemblies must be verified.

```
GROUP ID: V-225223
RULE ID: SV-225223r961038
```

Rationale:

A strong name consists of the assembly's identity, simple text name, version number, and culture information (if provided)—plus a public key and a digital signature. Strong names serve to identify the author of the code. If digital signatures used to sign strong name assemblies are not verified, any self signed code can be impersonated. This can lead to a loss of system integrity.

Audit:

Use regedit to review the Windows registry key

HKLM\Software\Microsoft\StrongName\Verification.

There should be no assemblies or hash values listed under this registry key. If the StrongName\Verification key does not exist, this is not a finding.

If there are assemblies or hash values listed in this key, each value represents a distinct application assembly that does not have the application strong name verified.

If any assemblies are listed as omitting strong name verification in a production environment, this is a finding.

If any assemblies are listed as omitting strong name verification in a development or test environment and the IAO has not provided documented approvals, this is a finding.

Remediation:

Use regedit to remove the values stored in Windows registry key HKLM\Software\Microsoft\StrongName\Verification. There should be no assemblies or hash values listed under this registry key.

All assemblies must require strong name verification in a production environment.

Strong name assemblies that do not require verification in a development or test environment must have documented approvals from the IAO.

Additional Information:

CCI-000185 For public key-based authentication, validate certificates by constructing and verifying a certification path to an accepted trust anchor including checking certificate status information.

- NIST SP 800-53::IA-5 (2)
- NIST SP 800-53A::IA-5 (2).1
- NIST SP 800-53 Revision 4::IA-5 (2) (a)
- NIST SP 800-53 Revision 5::IA-5 (2) (b) (1)

1.2 APPNET0046 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The Trust Providers Software Publishing State must be set to 0x23C00.

```
GROUP ID: V-225224
RULE ID: SV-225224r961038
```

Rationale:

Microsoft Windows operating systems provide a feature called Authenticode. Authenticode technology and its underlying code signing mechanisms serve to provide a structure to identify software publishers and ensure that software applications have not been tampered with. Authenticode technology relies on digital certificates and is based on Public Key Cryptography Standards (PKCS) #7 (encrypted key specification), PKCS #10 (certificate request formats), X.509 (certificate specification), and Secure Hash Algorithm (SHA) and MD5 hash algorithms.

The manner in which the Authenticode technology validates a certificate and determines what is considered a valid certificate can be modified to meet the mission of the Microsoft Windows system. Each facade of certificate validation is controlled through the bits that makeup the hexadecimal value for the Authenticode setting. An improper setting will allow non-valid certificates to be accepted and can put the integrity of the system into jeopardy.

The hexadecimal value of 0x23C00 will implement the following certificate enforcement policy:

- Trust the Test Root = FALSE
- Use expiration date on certificates = TRUE
- Check the revocation list = TRUE
- Offline revocation server OK (Individual) = TRUE
- Offline revocation server OK (Commercial) = TRUE
- Java offline revocation server OK (Individual) = TRUE
- Java offline revocation server OK (Commercial) = TRUE
- Invalidate version 1 signed objects = FALSE
- Check the revocation list on Time Stamp Signer = FALSE
- Only trust items found in the Trust DB = FALSE

Audit:

If the system or application being reviewed is SIPR based, this finding is NA.

This check must be performed for each user on the system.

Use regedit to locate "HKEY_USER[UNIQUE USER SID VALUE]\Software\Microsoft\Windows\CurrentVersion\WinTrust\Trust Providers\Software Publishing\State".

If the State value for any user is not set to the hexadecimal value of 0x23C00, this is a finding.

Remediation:

This fix must be performed for each user on the system.

Using regedit, change the hexadecimal value of the "HKEY_USER[UNIQUE USER SID VALUE]\Software\Microsoft\Windows\CurrentVersion\WinTrust\Trust Providers\Software Publishing\State" registry key to 0x23C00.

Additional Information:

CCI-000185 For public key-based authentication, validate certificates by constructing and verifying a certification path to an accepted trust anchor including checking certificate status information.

- NIST SP 800-53::IA-5 (2)
- NIST SP 800-53A::IA-5 (2).1
- NIST SP 800-53 Revision 4::IA-5 (2) (a)
- NIST SP 800-53 Revision 5::IA-5 (2) (b) (1)

1.3 APPNET0048 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Developer certificates used with the .NET Publisher Membership Condition must be approved by the ISSO.

```
GROUP ID: V-225225
RULE ID: SV-225225r961038
```

Rationale:

A .Net assembly will satisfy the Publisher Membership Condition if it is signed with a software publisher's Authenticode X.509v3 digital certificate that can be verified by the Windows operating system as having a chain of trust that leads to a trusted root certificate stored in the user's certificate store. The Publisher Membership Condition can be used to identify an organization, developer, vendor, or other entity as the ultimate source of the assembly, even if the code itself was obtained from a third party, such as a mirror site. Access to system resources, such as file systems or printers, may then be granted to the assembly based on the trust relationship with the identified entity.

Certificates used to sign assemblies so the Publisher Member Condition may be applied must originate from a trusted source. Using a certificate that is not from a trusted source could potentially violate system integrity and confidentiality.

Audit:

The infrastructure to enable Code Access Security (CAS) exists only in .NET Framework 2.x-4.x.

This requirement is Not Applicable (NA) for .NET Framework greater than 4.x.

(Note: The infrastructure is deprecated and is not receiving servicing or security fixes.)

Caspol.exe is a Microsoft tool used for working with .Net policy. Use caspol.exe to list the code groups and any publisher membership conditions.

The location of the caspol utility is dependent upon the system architecture of the system running .Net.

For 32 bit systems, caspol.exe is located at %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319.

For 64 bit systems, caspol.exe is located at %SYSTEMROOT%\Microsoft.NET\Framework64\v4.0.30319.

Example:

cd %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319

To check code groups for the machine, run the following command:

caspol.exe -m -lg

Sample Results:

Microsoft (R) .NET Framework CasPol 4.0.30319.1

Copyright (c) Microsoft Corporation. All rights reserved.

Policy change prompt is ON

Level = Machine

Code Groups:

1. All code: Nothing
2. 1.1. Zone - MyComputer: FullTrust (LevelFinal)
3. 1.1.1. StrongName - 00240000048000009400000006020000002400005253413100040000010001000 07D1FA57C4AED9F0A32E84AA0FAEFD0DE9E8FD6AEC8F87FB03766C834C 99921EB23BE79AD9D5DCC1DD9AD236132102900B723CF980957FC4E1771 08FC607774F29E8320E92EA05ECE4E821C0A5EFE8F1645C4C0C93C1AB99 285D622CAA652C1DFAD63D745D6F2DE5F17E5EAF0FC4963D261C8A12436 518206DC093344D5AD293: FullTrust
4. 1.1.2. StrongName - 0000000000000000040000000000000: FullTrust
5. 1.2. Zone - Intranet: LocalIntranet
6. 1.2.1. All code: Same site Web
7. 1.2.2. All code: Same directory FileIO - 'Read, PathDiscovery'
8. 1.3. Zone - Internet: Internet
9. 1.3.1. All code: Same site Web
10. 1.4. Zone - Untrusted: Nothing
11. 1.5. (First Match) Zone - Trusted: Internet
12. 1.5.1. All code: Same site Web
13. 1.6. Publisher - 30818902818100E47B359ACC061D70C237B572FA276C9854CFABD469DFB7 4E77D026630BEE2A0C2F8170A823AE69FDEB65704D7FD446DEFEF1F6BA1 2B6ACBDB1BFA7B9B595AB9A40636467CFF7C73F198B53A9A7CF177F6E78 96EBC591DD3003C5992A266C0AD9FBEE4E2A056BE7F7ED154D806F7965F 83B0AED616C192C6416CFCB46FC2F5CFD0203010001: FullTrust
14. Success

Section 1.6 above indicates the presence of a publisher's key that meets the Publisher's Membership Condition and is also given full trust.

If the Publisher Membership Condition is used on a nondefault Code Group and the use of that publisher's certificate is not documented and approved by the ISSO, this is a finding.

Remediation:

Trust must be established when utilizing Publishers Membership Condition. All publisher's certificates must have documented approvals from the ISSO.

Additional Information:

CCI-000185 For public key-based authentication, validate certificates by constructing and verifying a certification path to an accepted trust anchor including checking certificate status information.

- NIST SP 800-53::IA-5 (2)
- NIST SP 800-53A::IA-5 (2).1
- NIST SP 800-53 Revision 4::IA-5 (2) (a)
- NIST SP 800-53 Revision 5::IA-5 (2) (b) (1)

1.4 APPNET0052 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Encryption keys used for the .NET Strong Name Membership Condition must be protected.

```
GROUP ID: V-225226
RULE ID: SV-225226r961041
```

Rationale:

The Strong Name Membership condition requires that member assemblies be defined with Strong Names. A strong name consists of the assembly's identity, simple text name, version number, and culture information (if provided) — plus a public key and a digital signature. If assemblies do not have a strong name assigned, the assembly cannot be unique and the author of the code cannot be uniquely identified. In order to create the strong name, the developer must use a cryptographic key pair to sign the assembly. If the developer does not protect the key, the key can be stolen and used to sign any application, including malware applications. This could adversely affect application integrity and confidentiality.

Audit:

If the application is a COTS product, this requirement is Not Applicable (NA).

The infrastructure to enable Code Access Security (CAS) exists only in .NET Framework 2.x-4.x.

The requirement is Not Applicable (NA) for .NET Framework greater than 4.x.

(Note: The infrastructure is deprecated and is not receiving servicing or security fixes.)

Caspol.exe is a Microsoft tool used for working with .Net policy. Use caspol.exe to list the code groups and any publisher membership conditions.

The location of the caspol utility is dependent upon the system architecture of the system running .Net.

For 32 bit systems, caspol.exe is located at %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319.

For 64 bit systems, caspol.exe is located at %SYSTEMROOT%\Microsoft.NET\Framework64\v4.0.30319.

Example:

cd %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319

To check code groups, run the following command:

caspol.exe -all -lg

Sample response:

Microsoft (R) .NET Framework CasPol 4.0.30319.1

Security is ON

Execution checking is ON

Policy change prompt is ON

Level = Machine

Code Groups:

1. All code: Nothing
2. 1.1. Zone - MyComputer: FullTrust (LevelFinal)
3. 1.1.1. StrongName - 00240000048000009400000006020000002400005253413100040000010001000 07D1FA57C4AED9F0A32E84AA0FAEFD0DE9E8FD6AEC8F87FB03766C834C 99921EB23BE79AD9D5DCC1DD9AD236132102900B723CF980957FC4E1771 08FC607774F29E8320E92EA05ECE4E821C0A5EFE8F1645C4C0C93C1AB99 285D622CAA652C1DFAD63D745D6F2DE5F17E5EAF0FC4963D261C8A12436 518206DC093344D5AD293: FullTrust
4. 1.1.2. StrongName - 0000000000000000040000000000000000: FullTrust
5. 1.2. Zone - Intranet: LocalIntranet
6. 1.2.1. All code: Same site Web
7. 1.2.2. All code: Same directory FileIO - 'Read, PathDiscovery'
8. 1.3. Zone - Internet: Internet
9. 1.3.1. All code: Same site Web
10. 1.4. Zone - Untrusted: Nothing
11. 1.5. (First Match) Zone - Trusted: Internet
12. 1.5.1. All code: Same site Web
13. 1.6. Publisher - 30818902818100E47B359ACC061D70C237B572FA276C9854CFABD469DFB7 4E77D026630BEE2A0C2F8170A823AE69FDEB65704D7FD446DEFEF1F6BA1 2B6ACBDB1BFA7B9B595AB9A40636467CFF7C73F198B53A9A7CF177F6E78 96EBC591DD3003C5992A266C0AD9FBEE4E2A056BE7F7ED154D806F7965F 83B0AED616C192C6416CFCB46FC2F5CFD0203010001: FullTrust
14. Success

An assembly will satisfy the StrongNameMembershipCondition if its metadata contains the strongly identifying data associated with the specified strong name. At the least, this means it has been digitally signed with the private key associated with the public key recorded in the policy.
The presence of the encryption key values in the StrongName field indicates the use of StrongNameMembershipCondition.
If a Strong Name Membership Condition is assigned to a non-default Code Group the private key must be adequately protected by the software developer or the entity responsible for signing the assemblies.
Ask the Systems Programmer how the private keys are protected.
Private keys are simply values stored as strings of data. Keys can be stored in files on the file system or in a centralized data repository.
Adequate protection methods include, but are not limited to:

- utilizing centralized key management;
- using strict file permissions to limit access; and
- tying strong pass phrases to the key.

If the private key used to sign the assembly is not adequately protected, this is a finding.

Remediation:

Ask the Systems Programmer how the private keys used to sign the assembly are protected.

Private keys are simply values stored as strings of data. Keys can be stored in files on the file system or in a centralized data repository.

Adequate protection methods include, but are not limited to:

- utilizing centralized key management;
- using strict file permissions to limit access; and
- tying strong pass phrases to the key.

The private key(s) used to sign the assembly must be protected. Utilize centralized key management or strict file permissions along with strong pass phrases and/or other well-established industry practices for managing and controlling access to private keys.

Additional Information:

CCI-000186 For public key-based authentication, enforce authorized access to the corresponding private key.

- NIST SP 800-53::IA-5 (2)
- NIST SP 800-53A::IA-5 (2).1
- NIST SP 800-53 Revision 4::IA-5 (2) (b)
- NIST SP 800-53 Revision 5::IA-5 (2) (a) (1)

1.5 APPNET0055 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

CAS and policy configuration files must be backed up.

```
GROUP ID: V-225227
RULE ID: SV-225227r960936
```

Rationale:

A successful disaster recovery plan requires that CAS policy and CAS policy configuration files are identified and included in systems disaster backup and recovery events. Documentation regarding the location of system and application specific CAS policy configuration files and the frequency in which backups occur is required. If these files are not identified and the information is not documented, there is the potential that critical application configuration files may not be included in disaster recovery events which could lead to an availability risk.

Audit:

The infrastructure to enable Code Access Security (CAS) exists only in .NET Framework 2.x-4.x.

The requirement is Not Applicable (NA) for .NET Framework greater than 4.x.

(Note: The infrastructure is deprecated and is not receiving servicing or security fixes.)

Ask the System Administrator if all CAS policy and policy configuration files are included in the system backup. If they are not, this is a finding.

Ask the System Administrator if the policy and configuration files are backed up prior to migration, deployment, and reconfiguration. If they are not, this is a finding.

Ask the System Administrator for documentation that shows CAS Policy configuration files are backed up as part of a disaster recovery plan. If they have no documentation proving the files are backed up, this is a finding.

Remediation:

All CAS policy and policy configuration files must be included in the system backup.

All CAS policy and policy configuration files must be backed up prior to migration, deployment, and reconfiguration.

CAS policy configuration files must be included in disaster recovery plan documentation.

Additional Information:

CCI-000164 Protect audit information from unauthorized deletion.

- NIST SP 800-53::AU-9
- NIST SP 800-53A::AU-9.1
- NIST SP 800-53 Revision 4::AU-9
- NIST SP 800-53 Revision 5::AU-9 a

1.6 APPNET0060 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Remoting Services HTTP channels must utilize authentication and encryption.

```
GROUP ID: V-225228
RULE ID: SV-225228r1043178
```

Rationale:

Note: Microsoft recommends using the Windows Communication Framework (WCF) rather than using .Net remoting. New development projects should refrain from using .Net remoting capabilities whenever possible.

.NET remoting provides the capability to build widely distributed applications. The application components may reside all on one computer or they may be spread out across the enclave. .NET client applications can make remoting calls to use objects in other processes on the same computer or on any other computer that is reachable over the network. .NET remoting can also be used to communicate with other application domains within the same process. Remoting is achieved via the exposure of endpoints that can be used to establish remote connectivity.

Normally when application code attempts to access a protected resource, a stack walk is performed to ensure that all stack frames have permission to access the resource. However, with .Net 4.0, when a call is made on a remote object, this stack walk is not performed across the remoting boundary. The .Net remoting infrastructure requires FullTrust permission to execute on either the client or the server.

Due to the fact that FullTrust permission is required, Remoting endpoints should be authenticated and encrypted in order to protect the system and the data.

Microsoft provides three different "channels" that are used for remoting. They are HTTP, TCP, and IPC.

Any unauthorized use of a remoting application provides unauthorized access with FullTrust permissions to the system. This can potentially result in a loss of system integrity or confidentiality.

Audit:

If .NET remoting with HTTP channel is not used, this check is Not Applicable.

Review the machine.config file and the [application name].exe.config file.

For 32-bit systems, the "machine.config" file is contained in the following folder:
%SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319\Config

For 64-bit systems, the "machine.config" file is contained in the following folder:
%SYSTEMROOT%\Microsoft.NET\Framework64\v4.0.30319\Config.

Microsoft specifies locating the [application].config file in the same folder as the application executable (.exe) file. However, the developer does have the capability to specify a different location when the application is compiled. Therefore, if the file is not found in the application home folder, a search of the system is required. If the [application name].exe.config file is not found on the system, then only a check of the machine.config file is required.

Sample machine/application config file:

Microsoft provides three "channels" that are used for remoting connectivity. They are the HTTP, TCP, and IPC channels. The channel that is used is specified via the element in the config file.

HTTP channel example:

<channel ref=“http server” port=“80”/>

The HTTP channel only supports encryption and message integrity when the remote object is hosted in Internet Information Services (IIS) using TLS.

The above example shows the well-known TLS port of 443 is not being used.

If the HTTP remoting channel is not configured to protect the channel by using TLS encryption, this is a finding.

Remediation:

If .NET remoting with HTTP channel is not used, this fix is Not Applicable.

Ensure encryption and message integrity are used for HTTP remoting channels.

The HTTP channel only supports encryption and message integrity when the remote object is hosted in Internet Information Services (IIS) using TLS.

HTTP channels are protected via TLS (HTTPS).

<channel ref=“http server” port=“443”/>

Change the channel ref parameter to utilize a TLS port and leverage TLS on the remote IIS server.

Additional Information:

CCI-001184 Protect the authenticity of communications sessions.

- NIST SP 800-53::SC-23
- NIST SP 800-53A::SC-23.1
- NIST SP 800-53 Revision 4::SC-23
- NIST SP 800-53 Revision 5::SC-23

1.7 APPNET0061 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

.Net Framework versions installed on the system must be supported.

```
GROUP ID: V-225229
RULE ID: SV-225229r961863
```

Rationale:

Unsupported software introduces risks and violates DoD policy. Applications utilizing unsupported versions of .NET introduce substantial risk to the host, network, and the enclave by virtue of the fact they leverage an architecture that is no longer updated by the vendor. This introduces potential application integrity, availability, or confidentiality issues.

Audit:

Determine which versions of the .NET Framework are installed by opening the directory %systemroot%\Microsoft.NET.

The folder named "%systemroot%\Microsoft.NET\Framework" contains .NET files for 32 bit systems. The folder named "%systemroot%\Microsoft.NET\Framework64" contains .NET files for 64 bit systems. 64 bit systems will have both the 32 bit and the 64 bit folders while 32 bit systems do not have a Framework64 folder.

Within each of the aforementioned folders are the individual folder names that contain the corresponding versions of the .NET Framework:

v4.0.30319

v3.5

v3.0

v2.0.50727

v1.1.4322

v1.0.3705

Search for all the Mscorlib.dll files in the %systemroot%\Microsoft.NET\Framework folder and the %systemroot%\Microsoft.NET\Framework64 folder if the folder exists. Click on each of the files, view properties, and click version tab to determine the version installed. If there is no Mscorlib.dll, there is no installed version of .Net Framework in that directory.

More specific information on determining versions of .Net Framework installed can be found at the following link. http://support.microsoft.com/kb/318785

Verify extended support is available for the installed versions of .Net Framework.

Verify the .Net Framework support dates with Microsoft Product Lifecycle Search link.

http://support.microsoft.com/lifecycle/search/?sort=PN&alpha=.NET+Framework

Beginning with .NET 3.5 SP1, the .NET Framework is considered a Component of the Windows OS. Components follow the Support Lifecycle policy of their parent product or platform.

If any versions of the .Net Framework are installed and support is no longer available, this is a finding.

Remediation:

Remove unsupported versions of the .NET Framework and upgrade legacy applications that utilize unsupported versions of the .NET framework.

Additional Information:

CCI-000366 Implement the security configuration settings.

- NIST SP 800-53::CM-6 b
- NIST SP 800-53A::CM-6.1 (iv)
- NIST SP 800-53 Revision 4::CM-6 b
- NIST SP 800-53 Revision 5::CM-6 b

1.8 APPNET0062 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

The .NET CLR must be configured to use FIPS approved encryption modules.

```
GROUP ID: V-225230
RULE ID: SV-225230r961908
```

Rationale:

FIPS encryption is configured via .NET configuration files. There are numerous configuration files that affect different aspects of .Net behavior. The .NET config files are described below.

Machine Configuration Files: The machine configuration file, Machine.config, contains settings that apply to an entire computer. This file is located in the %SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319\Config directory for 32 bit .NET 4 installations and %SYSTEMROOT%\Microsoft.NET\Framework64\v4.0.30319\Config for 64 bit systems. Machine.config contains configuration settings for machine-wide assembly binding, built-in remoting channels, and ASP.NET.

Application Configuration Files: Application configuration files contain settings specific to an application. If checking these files, a .NET review of a specific .NET application is most likely being conducted. These files contain configuration settings that the Common Language Runtime reads (such as assembly binding policy, remoting objects, and so on), and settings that the application can read.

The name and location of the application configuration file depends on the application's host, which can be one of the following:

Executable–hosted application configuration files.

The configuration file for an application hosted by the executable host is in the same directory as the application. The name of the configuration file is the name of the application with a .config extension. For example, an application called myApp.exe can be associated with a configuration file called myApp.exe.config.

Internet Explorer-hosted application configuration files.

If an application hosted in Internet Explorer has a configuration file, the location of this file is specified in a tag with the following syntax.

In this tag, "location" represents a URL that point to the configuration file. This sets the application base. The configuration file must be located on the same web site as the application.

.NET 4.0 allows the CLR runtime to be configured to ignore FIPS encryption requirements. If the CLR is not configured to use FIPS encryption modules, insecure encryption modules might be employed which could introduce an application confidentiality or integrity issue.

Audit:

Examine the .NET CLR configuration files from the vulnerability discussion to find the runtime element and then the "enforceFIPSPolicy" element.

Example:

<enforceFIPSPolicy enabled="true|false" />

By default, the .NET "enforceFIPSPolicy" element is set to "true".

If the "enforceFIPSPolicy" element does not exist within the "runtime" element of the CLR configuration, this is not a finding.

If the "enforceFIPSPolicy" element exists and is set to "false", and the IAO has not accepted the risk and documented the risk acceptance, this is a finding.

Remediation:

Examine the .NET CLR configuration files to find the runtime element and then the "enforceFIPSPolicy" element.

Example:

<enforceFIPSPolicy enabled="true|false" />

Delete the "enforceFIPSPolicy" runtime element, change the setting to "true" or there must be documented IAO approvals for the FIPS setting.

Additional Information:

CCI-002450 Implement organization-defined types of cryptography for each specified cryptography use.

- NIST SP 800-53 Revision 4::SC-13
- NIST SP 800-53 Revision 5::SC-13 b

1.9 APPNET0063 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

.NET must be configured to validate strong names on full-trust assemblies.

```
GROUP ID: V-225231
RULE ID: SV-225231r961038
```

Rationale:

The "bypassTrustedAppStrongNames" setting specifies whether the bypass feature that avoids validating strong names for full-trust assemblies is enabled. By default the bypass feature is enabled in .Net 4, therefore strong names are not validated for correctness when the assembly/program is loaded. Not validating strong names provides a faster application load time but at the expense of performing certificate validation.

Full trust assemblies are .Net applications launched from the local host. Strong names are digital signatures tied to .Net applications/assemblies. .Net 4 considers applications installed locally to be fully trusted by default and grants these applications full permissions to access host resources.

The bypass feature applies to any assembly signed with a strong name and having the following characteristics:

```
Fully trusted without the StrongName evidence (for example, has MyComputer
zone evidence).

Loaded into a fully trusted AppDomain.

Loaded from a location under the ApplicationBase property of that AppDomain.

Not delay-signed.
```

Not validating the certificates used to sign strong name assemblies will provide a faster application load time, but falsely assumes that signatures used to sign the application are to be implicitly trusted. Not validating strong name certificates introduces an integrity risk to the system.

Audit:

If there is documented ISSO risk acceptance for development systems, this is not a finding.

For 32 bit production systems:

Use regedit to examine the “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework” key.

On 64-bit production systems:

Use regedit to examine both the “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework” and “HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft.NETFramework” keys.

If the "AllowStrongNameBypass" value does not exist, or if the “DWORD” value is set to “1”, this is a finding.

Documentation must include a complete list of installed .Net applications, application versions, and acknowledgement that ISSO trusts each installed application.

If application versions installed on the system do not match approval documentation, this is a finding.

Remediation:

For 32 bit production systems:

Set “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\AllowStrongNameBypass" to a “DWORD” value of “0”.

On 64-bit production systems:

Set “HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\ AllowStrongNameBypass” and “HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft.NETFramework\ AllowStrongNameBypass” to a “DWORD” value of “0”.

Or, obtain documented ISSO risk acceptance for each .Net application installed on the system.

Approval documentation will include complete list of all installed .Net applications, application versions, and acknowledgement of ISSO trust of each installed application.

Additional Information:

CCI-000185 For public key-based authentication, validate certificates by constructing and verifying a certification path to an accepted trust anchor including checking certificate status information.

- NIST SP 800-53::IA-5 (2)
- NIST SP 800-53A::IA-5 (2).1
- NIST SP 800-53 Revision 4::IA-5 (2) (a)
- NIST SP 800-53 Revision 5::IA-5 (2) (b) (1)

1.10 APPNET0066 (Manual)

Profile Applicability:

- SEVERITY: CAT III

Description:

.NET default proxy settings must be reviewed and approved.

```
GROUP ID: V-225234
RULE ID: SV-225234r961863
```

Rationale:

The .Net framework can be configured to utilize a different proxy or altogether bypass the default proxy settings in the client's browser. This may lead to the framework using a proxy that is not approved for use. If the proxy is malicious, this could lead to a loss of application integrity and confidentiality.

Audit:

Open Windows explorer and search for all "*.exe.config" and "machine.config" files.

Search each file for the "defaultProxy" element.

<defaultProxy

enabled="true|false"

useDefaultCredentials="true|false"

…

…

…

/>

If the "defaultProxy" setting "enabled=false" or if the "bypasslist", "module", or "proxy" child elements have configuration entries and there are no documented approvals from the IAO, this is a finding.

If the "defaultProxy" element is empty or if "useSystemDefault =True” then the framework is using default browser settings, this is not a finding.

Remediation:

Open Windows explorer and search for all "*.exe.config" and "machine.config" files.

Search each file for the "defaultProxy" element.

Clear the values contained in the "defaultProxy" element, and the "bypasslist", "module", and "proxy" child elements.

The IAO must provide documented approvals of any non-default proxy servers.

Additional Information:

CCI-000366 Implement the security configuration settings.

- NIST SP 800-53::CM-6 b
- NIST SP 800-53A::CM-6.1 (iv)
- NIST SP 800-53 Revision 4::CM-6 b
- NIST SP 800-53 Revision 5::CM-6 b

1.11 APPNET0067 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Event tracing for Windows (ETW) for Common Language Runtime events must be enabled.

```
GROUP ID: V-225235
RULE ID: SV-225235r960891
```

Rationale:

Event tracing captures information about applications utilizing the .NET CLR and the .NET CLR itself. This includes security oriented information, such as Strong Name and Authenticode verification.

Beginning with Windows Vista, ETW is enabled by default however, the .Net CLR and .Net applications can be configured to not utilize Event Tracing. If ETW event tracing is disabled, critical events that occurred within the runtime will not be captured in event logs.

Audit:

Open Windows explorer and search for all .NET config files including application config files (*.exe.config)

NOTE:

Beginning with Windows Vista and Windows Server 2008, ETW Tracing is enabled by default and the "etwEnable" setting is not required in order for Event Tracing to be enabled. An etwEnable setting of "true" IS required in earlier versions of Windows as ETW is disabled by default.

Examine the configuration settings for

<etwEnable enabled="false" />.

If the "etwEnable" element is set to "true", this is not a finding.

If the "etwEnable" element is set to "false" and documented approvals by the IAO are not provided, this is a finding.

Remediation:

Open Windows explorer and search for all .NET config files including application config files (*.exe.config).

Examine the configuration settings for

<etwEnable enabled="false" />.

Enable ETW Tracing by setting the etwEnable flag to "true" or obtain documented IAO approvals.

Additional Information:

CCI-000130 Ensure that audit records contain information that establishes what type of event occurred.

- NIST SP 800-53::AU-3
- NIST SP 800-53A::AU-3.1
- NIST SP 800-53 Revision 4::AU-3
- NIST SP 800-53 Revision 5::AU-3 a

1.12 APPNET0064 (Manual)

Profile Applicability:

- SEVERITY: CAT III

Description:

.Net applications that invoke NetFx40_LegacySecurityPolicy must apply previous versions of .NET STIG guidance.

```
GROUP ID: V-225232
RULE ID: SV-225232r1050651
```

Rationale:

CAS policy is .NET runtime version-specific. In .NET Framework version 4, CAS policy is disabled by default however; it can be re-enabled by using the NetFx40_LegacySecurityPolicy setting on a per application basis. Caspol.exe is provided by Microsoft to set security policy on .Net applications prior to version 4.0. This requirement does not apply to the caspol.exe assembly or other assemblies provided with the Windows OS or the Windows Secure Host Baseline (SHB).

When invoking the NetFx40_LegacySecurityPolicy setting in .NET 4, earlier versions of the .NET Framework CAS policy will become active therefore previous .NET STIG guidance that applies to the reactivated versions must also be applied.

Failure to apply applicable versions of STIG guidance can result in the loss of system confidentiality, integrity or availability.

Audit:

The infrastructure to enable Code Access Security (CAS) exists only in .NET Framework 2.x - 4.x.

The requirement is Not Applicable (NA) for .NET Framework > 4.x.

(Note: The infrastructure is deprecated and is not receiving servicing or security fixes.)

Open Windows explorer and search for all *.exe.config files. This requirement does not apply to the caspol.exe assembly or other assemblies provided with the Windows OS or the Windows Secure Host Baseline (SHB).

To find relevant files, run the FINDSTR command from an elevated (admin) command prompt:

FINDSTR /i /s "NetFx40_LegacySecurityPolicy" c:*.exe.config

This command will search all ."exe.config" files on the c: drive partition for the "LegacySecurityPolicy" setting. Repeat the command for each drive partition on the system.

If the .NET application configuration file uses the legacy policy element, and .NET STIG guidance that covers these legacy versions has not been applied, this is a finding.

Remediation:

The infrastructure to enable CAS exists only in .NET Framework 2.x - 4.x.

The requirement is Not Applicable (NA) for .NET Framework > 4.x.

(Note: The infrastructure is deprecated and is not receiving servicing or security fixes.)

Apply the .NET Framework Security Checklist for .Net versions 1 through 3.5 when using the NetFx40_LegacySecurityPolicy setting.

Additional Information:

CCI-000366 Implement the security configuration settings.

- NIST SP 800-53::CM-6 b
- NIST SP 800-53A::CM-6.1 (iv)
- NIST SP 800-53 Revision 4::CM-6 b
- NIST SP 800-53 Revision 5::CM-6 b

1.13 APPNET0065 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Trust must be established prior to enabling the loading of remote code in .Net 4.

```
GROUP ID: V-225233
RULE ID: SV-225233r961608
```

Rationale:

In the .NET Framework version 3.5 and earlier versions, if an application assembly loaded code/objects from a remote location, that assembly would run partially trusted with a permissions grant set that depended on the zone in which it was loaded. For example, if an assembly was loaded from a web site, it was loaded into the Internet zone and granted the Internet permission set. In other words, it was executed in an Internet sandbox.

If the same program is run in the .NET Framework version 4, an exception is thrown which effectively states; either explicitly create a sandbox for the assembly or run it in full trust.

The <loadFromRemoteSources> element specifies the assemblies that run partially trusted in earlier versions of the .NET Framework will be run fully trusted in the .NET Framework 4.

If loadFromRemoteSources is set to true, the remotely loaded application code is granted full trust. This could create an integrity vulnerability on the system. The required method to address this is to explicitly create a sandboxed environment for the remotely loaded code to run in rather than allowing remotely loaded code to run with full trust.

The appropriate level of trust must be established prior to enabling the loading of remote code in .Net 4 applications and that code must be run in a controlled environment. The following is an example of the use of loadFromRemoteSources.

Audit:

Open Windows explorer and search for *.exe.config.

Search each config file found for the "loadFromRemoteSources" element.

If the loadFromRemoteSources element is enabled

("loadFromRemoteSources enabled = true"), and the remotely loaded application is not run in a sandboxed environment, or if OS based software controls, such as AppLocker or Software Security Policies, are not utilized, this is a finding.

Remediation:

.Net application code loaded from a remote source must be run in a controlled environment.

A controlled environment consists of a sandbox, such as running in an Internet Explorer host environment or employing OS based software access controls, such as AppLocker or Software Security Policies, when application design permits.

Obtain documented IAO approvals for all remotely loaded code.

Additional Information:

CCI-002530 Maintain a separate execution domain for each executing system process.

- NIST SP 800-53 Revision 4::SC-39
- NIST SP 800-53 Revision 5::SC-39

1.14 APPNET0070 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Software utilizing .Net 4.0 must be identified and relevant access controls configured.

```
GROUP ID: V-225236
RULE ID: SV-225236r1069477
```

Rationale:

With the advent of .Net 4.0, the .Net framework no longer directly configures or enforces security policy for .Net applications. This task is now relegated to the operating system layer and the security protections built-in to .Net application "runtime hosts" that run on the O.S.

Examples of these .Net "runtime hosts" include; Internet Explorer, Windows Shell, ASP.NET, Database Engines or any other "runtime hosts" that utilize .Net and load the .Net CLR.

Security protections include utilizing runtime host security controls such as sandboxing to restrict or control application behavior as designed or required.

To compensate for these design changes, Windows provides native solutions such as Software Security Policies (SSP) and Application Locker (AL) which are technologies that can be implemented via Group Policy (GPO). SSP, AL and similar third party solutions serve to restrict execution of applications, scripts and libraries based upon cryptographic hash, security zones, path and certificate values that are associated with the application files. Additionally, application developers will utilize "sandboxing" techniques within their code in order to isolate 3rd party code libraries from critical system resources.

In order to assign protections to .Net 4.0 applications, the applications must first be identified and the appropriate hosting security mechanisms configured to accomplish that task.

.Net STIG guidance cannot be applied if .Net applications are not identified and documented. The lack of an application inventory introduces confidentiality, availability and integrity vulnerabilities to the system.

Audit:

This requirement does not apply to the "caspol.exe" assembly or other assemblies provided with the Windows OS or the Windows Secure Host Baseline (SHB).

Ask the system administrator to provide documentation that identifies:

- Each .Net 4.0 application run on the system.
- The .Net runtime host that invokes the application.
- The security measures employed to control application access to system resources or user access to application.

For additional insight run: tasklist /fi "modules eq mscoree.dll"
If all .Net applications, runtime hosts and security protections have been documented or if there are no .Net 4.0 applications existing on the system, this is not a finding.
If there is no documentation that identifies the existence of .NET 4.0 applications or the lack thereof, this is a finding.
If the runtime hosts have not been identified, this is a finding.
If the security protections have not been identified, this is a finding.

Remediation:

Document the existence of all .Net 4.0 applications that are not provided by the host Windows OS or the Windows Secure Host Baseline (SHB).

Document the corresponding runtime hosts that are used to invoke the applications.

Document the applications security control requirements (restricting application access to resources or user access to the application).

Additional Information:

CCI-002530 Maintain a separate execution domain for each executing system process.

- NIST SP 800-53 Revision 4::SC-39
- NIST SP 800-53 Revision 5::SC-39

1.15 APPNET0071 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Remoting Services TCP channels must utilize authentication and encryption.

```
GROUP ID: V-225237
RULE ID: SV-225237r1043178
```

Rationale:

Note: Microsoft recommends using the Windows Communication Framework (WCF) rather than .Net remoting. New development projects should refrain from using .Net remoting capabilities whenever possible.

.NET remoting provides the capability to build widely distributed applications. The application components may reside all on one computer or they may be spread out across the enclave. .NET client applications can make remoting calls to use objects in other processes on the same computer or on any other computer that is reachable over the network. .NET remoting can also be used to communicate with other application domains within the same process. Remoting is achieved via the exposure of endpoints that can be used to establish remote connectivity.

Normally when application code attempts to access a protected resource, a stack walk is performed to ensure that all stack frames have permission to access the resource. However, with .Net 4.0, when a call is made on a remote object, this stack walk is not performed across the remoting boundary. The .Net remoting infrastructure requires FullTrust permission to execute on either the client or the server.

Due to the fact that FullTrust permission is required, Remoting endpoints should be authenticated and encrypted in order to protect the system and the data.

Microsoft provides three different "channels" that are used for remoting. They are HTTP, TCP, and IPC.

Any unauthorized use of a remoting application provides unauthorized access with FullTrust permissions to the system. This can potentially result in a loss of system integrity or confidentiality.

Audit:

If .NET remoting with TCP channel is not used, this check is Not Applicable.

Check the machine.config and the [application executable name].exe.config configuration files.

For 32-bit systems, the "machine.config" file is contained in the following folder.
%SYSTEMROOT%\Microsoft.NET\Framework\v4.0.30319\Config

For 64-bit systems, the "machine.config" file is contained in the following folder.
%SYSTEMROOT%\Microsoft.NET\Framework64\v4.0.30319\Config.

Microsoft specifies locating the application config file in the same folder as the application executable (.exe) file. However, the developer does have the capability to specify a different location when the application is compiled. Therefore, if the config file is not found in the application home folder, a search of the system is required. If the [application name].exe.config file is not found on the system, then only a check of the machine.config file is required.

Sample machine/application config file:

Microsoft provides three "channels" that are used for remoting connectivity. They are the HTTP, TCP, and IPC channels. The channel that is used is specified via the element in the config file.

TCP channel example:

The TCP channel provides encryption and message integrity when the "secure" flag is set to "true" as shown in the above example.

If the "secure" flag is not set to "true" for the TCP channel, this is a finding.

Remediation:

If .NET remoting with TCP channel is not used, this fix is Not Applicable.

Ensure encryption and message integrity are used for TCP remoting channels.

TCP remoting connections are protected via the secure=true configuration parameter.

Include the secure="true" flag in the channel ref parameter of the machine.config and [application name].exe.config file if the [application name].exe.config file exists on the system.

Additional Information:

CCI-001184 Protect the authenticity of communications sessions.

- NIST SP 800-53::SC-23
- NIST SP 800-53A::SC-23.1
- NIST SP 800-53 Revision 4::SC-23
- NIST SP 800-53 Revision 5::SC-23

1.16 APPNET0075 (Manual)

Profile Applicability:

- SEVERITY: CAT II

Description:

Update and configure the .NET Framework to support TLS.

```
GROUP ID: V-225238
RULE ID: SV-225238r1069480
```

Rationale:

Use of the RC4 cipher in TLS could allow an attacker to perform man-in-the-middle attacks and recover plaintext from encrypted sessions. Applications that target .Net version 4.x running on multiple Windows versions could be vulnerable to these types of attacks. The registry settings in this requirement will prevent .Net applications that target the 4.x framework from selecting and utilizing the Schannel.dll RC4 cipher for TLS connections. Applications that use TLS when connecting to remote systems will perform a handshake and negotiate the TLS version and cipher that is to be used between the client and the server. This is standard protocol for all TLS connections. If the server and client are not configured to use the same TLS version and cipher, the TLS connection may fail. Applications should be tested with these registry settings prior to production implementation of the fix in order to avoid application outages.

Audit:

In older Windows systems (Windows Server 2012 or earlier), TLS 1.2 must be enabled systemwide by setting "SchUseStrongCrypto".

SystemDefaultTlsVersions is a configuration switch in .NET Framework (starting from 4.6) that allows the application to use the default TLS version supported by the underlying Windows operating system instead of hardcoding a specific TLS version (like TLS 1.2).

Check Registry:

Use regedit to review the following Windows registry keys:

For 32-bit systems:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\v4.0.30319\

For 64 bit systems:

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\v4.0.30319

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft.NETFramework\v4.0.30319\

1. If the "SchUseStrongCrypto" value name does not exist, or is not a REG_DWORD type set to "1", this is a finding.
2. For .NET Framework >4.6, use the default TLS version supported by the underlying Windows operating system.
3. If the "SystemDefaultTlsVersions" value name does not exist, or is not a REG_DWORD type set to "1", this is a finding.

Note: The SchUseStrongCrypto setting allows .NET to use TLS 1.1 and TLS 1.2. The SystemDefaultTlsVersions setting allows .NET to use the OS configuration.

Remediation:

1. SchUseStrongCrypto enabled:

Use regedit to access the following registry key.
For 32-bit systems:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\v4.0.30319\
For 64-bit systems:
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\v4.0.30319

HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft.NETFramework\v4.0.30319\
Modify or create the following Windows registry value: SchUseStrongCrypto.
Set SchUseStrongCrypto to a REG_DWORD value of "1".

2. SystemDefaultTlsVersions enabled (.NET Framework >4.6):
3. For 64-bit Windows, create a .reg file with the following content and apply it:
4. Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft.NETFramework\v4.0.30319]
"SystemDefaultTlsVersions"=dword:00000001
[HKEY_LOCAL_MACHINE\SOFTWARE\Wow6432Node\Microsoft.NETFramework\v4.0.30319]
"SystemDefaultTlsVersions"=dword:00000001

3. Restart the system for changes to take effect.

Additional Information:

CCI-001762 Disable or remove organization-defined functions, ports, protocols, software, and services within the system deemed to be unnecessary and/or nonsecure.

- NIST SP 800-53 Revision 4::CM-7 (1) (b)
- NIST SP 800-53 Revision 5::CM-7 (1) (b)

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**STIG RULES**		
1.1	APPNET0031 (Manual)	☐	☐
1.2	APPNET0046 (Manual)	☐	☐
1.3	APPNET0048 (Manual)	☐	☐
1.4	APPNET0052 (Manual)	☐	☐
1.5	APPNET0055 (Manual)	☐	☐
1.6	APPNET0060 (Manual)	☐	☐
1.7	APPNET0061 (Manual)	☐	☐
1.8	APPNET0062 (Manual)	☐	☐
1.9	APPNET0063 (Manual)	☐	☐
1.10	APPNET0066 (Manual)	☐	☐
1.11	APPNET0067 (Manual)	☐	☐
1.12	APPNET0064 (Manual)	☐	☐
1.13	APPNET0065 (Manual)	☐	☐
1.14	APPNET0070 (Manual)	☐	☐
1.15	APPNET0071 (Manual)	☐	☐
1.16	APPNET0075 (Manual)	☐	☐

Appendix: Change History

Date	Version	Changes for this version
Sep 3, 2025	1.0.0	Initial CIS Release