# App Defense Alliance Mobile Application Specification

Version 1.0 - 10-OCT 24

## Revision History

| Version | Date | Description |
|---|---|---|
| 0.5 | 10-MAY 24 | Initial draft based on Mobile App Tiger Team review of MASVS specification |
| 0.7 | 25-MAY 24 | Updates from Tiger Team review of 0.5 spec |
| 0.9 | 9-AUG 24 | Updates from ASA WG leads review of 0.7 spec |
| 1.0 | 10-OCT 24 | Approved for release by the alliance steering committee |

## Acknowledgements

The App Defense Alliance Application Security Assessment Working Group (ASA WG) would like to thank the following individuals for their contributions to this specification.

### Application Security Assessment Working Group Leads

- Alex Duff (Meta) - ASA WG Chair
- Brooke Davis (Google) - ASA WG Vice Chair

### Mobile Profile Leads

- Brooke Davis (Google)
- Tim Bolton (Meta)

### Contributors

- Alex Duff (Meta)
- Ana Vargas
- Anna Bhirud (Google)
- Antonio Nappa (Zimperium)
- Anushree Shetty (KPMG)
- Artem Chornyi
- Artur Gartvikh
- Asaf Peleg (Zimperium)
- Bhairavi Mehta (TAC Security)
- Brad Ree (Google)
- Brooke Davis (Google)
- Carlos Holguera
- Chris Cinnamo (Zimperium)
- Christian Schnell (Zimperium)
- Cody Martin (Leviathan Security)
- Corey Gagnon (Meta)

- Eugene Liderman (Google)
- Gianluca Braga (Zimperium)
- Joel Scambray (NCC Group)
- Jon Paterson (Zimperium)
- Jorge Damian
- Jorge Wallace Ruiz (Dekra)
- José María Santos López
- Juan Manuel Martinez Hernandez
- Julia McLaughlin (Google)
- Jullian Gerhart (NCC Group)
- Kelly Albrink (Bishop Fox)
- Mamachan Anish (KPMG)
- Mark Stribling (Leviathan Security)
- Mateo Morales Amador
- Michael Krueger
- Michael Whiteman (Meta)
- Nazariy Haliley (Bishop Fox)
- Nicole Weisenbach (NCC Group)
- Noelle Murata (Leviathan Security)
- Olivier Tuchon
- Pamela Dingle (Microsoft)
- Rubén Lirio (Dekra)
- Rupesh Nair (Net Sentries)
- Sebastian Porst
- Shad Malloy
- Soledad Antelada Toledano (Google)
- Syrone Hanks II
- Thomas Cannon (NCC Group)
- Tim Bolton (Meta)
- Viktor Sytnik (Leviathan Security)
- Yiannis Kozyrakis
- Zach Moreno (Bishop Fox)

# Introduction

In today’s digitally-driven world, mobile applications are the backbone of countless businesses and organizations. Unfortunately, they are also prime targets for cyberattacks that threaten data confidentiality, service availability, and overall business integrity. To mitigate risks and build a secure mobile environment, a robust mobile application security standard and certification program is essential.

## Our Approach: OWASP MASVS as the Foundation

This program leverages the internationally recognized OWASP Mobile Application Security Verification Standard (MASVS) as its core. The OWASP MASVS offers a comprehensive set of security assessment requirements and guidelines covering the entire mobile application development lifecycle. Building upon this base, the App Defense Alliance (ADA) focused on testable requirements with clear acceptance criteria. Further, the ADA approach emphasizes the use of automation where possible.

## Applicability

This document is intended for system and application administrators, security specialists, auditors, help desk, platform deployment, and/or DevOps personnel who plan to develop, deploy, assess, or secure mobile applications.

## References

1. OWASP Mobile Application Security Verification Standard

## Licensing



## Assumptions

The following assumptions are intended to aid the Authorized Labs for baseline security testing.

**Platform**

The mobile application relies upon a trustworthy computing platform that runs a recent version of a mobile operating system (i.e. N-2) from the date of evaluation. For the purposes of this document, N refers to a major operation system release.

**Proper User**

The user of the application software is not willfully negligent or hostile, and sets a device PIN/Passcode.

**Sensitive or Confidential Data**

Data that is of particular concern from a security perspective, including user data, user device data, company data, credentials, keys, or other types of confidential information. Throughout this document, the phrase "sensitive data" refers to these kinds of data and should not be confused with the meaning of *Sensitive Data* under regulations like GDPR or other regulatory regimes.

Note that apps in certain verticals such as healthcare or finance may have to meet higher security, privacy, and regulatory requirements.

**Tooling**

The ADA approach emphasizes the use of automation where possible. We expect future tooling investment to assist with gathering of developer evidence for Level 1 assurance.

## Definitions

| Term | Definition |
| --- | --- |

| Term | Definition |
| --- | --- |
| (AL1) ADA Assurance Level 1 (Verified Self Assessment) | The developer provides evidence and statements of compliance to each audit test case. The ADA approved lab reviews the evidence against the requirements. The ADA approved lab does not directly assess the application. |
| (AL2) ADA Assurance Level 2 (Lab Assessment) | The ADA approved lab evaluates each audit test case directly against the application. In some cases, the developer may need to provide limited information or code snippets. |

# Table of Contents

# 1 ANDROID

## 1.1 Storage

### 1.1.1 The app securely stores sensitive data in external storage

**Description**

This control ensures that any sensitive data that is intentionally stored by the app is properly protected independently of the target location.

**Rationale**

Apps handle sensitive data coming from many sources such as the user, the backend, system services or other apps on the device and usually need to store it locally. The storage locations may be private to the app (e.g. its internal storage) or be public and therefore accessible by the user or other installed apps (e.g. public folders such as Downloads).

**Audit**

| Spec | Description |
|---|---|
| 1.1.1.1 | The app shall securely store sensitive data in external storage |

### 1.1.2 The app prevents leakage of sensitive data

**Description**

This control covers unintentional data leaks where the developer actually has a way to prevent it.

**Rationale**

There are cases when sensitive data is unintentionally stored or exposed to publicly accessible locations; typically as a side-effect of using certain APIs, system capabilities such as backups or logs.

**Audit**

| Spec | Description |
|---|---|
| 1.1.2.1 | The Keyboard Cache shall be disabled for sensitive data inputs |
| 1.1.2.2 | No sensitive data shall be stored in system logs |

## 1.2 Crypto

### 1.2.1 The app employs current strong cryptography and uses it according to industry best practices

**Description**

This control covers general cryptography best practices, which are typically defined in external standards. For testing, the Crypto requirements only apply to sensitive data stored outside of the application sandbox.

**Rationale**

Cryptography plays an especially important role in securing the user's data - even more so in a mobile environment, where attackers having physical access to the user's device is a likely scenario.

**Audit**

| Spec | Description |
|---|---|
| 1.2.1.1 | No insecure random number generators shall be utilized for any security sensitive context |
| 1.2.1.2 | No insecure operations shall be used for symmetric cryptography |
| 1.2.1.3 | Strong cryptography shall be implemented according to industry best practices |

### 1.2.2 The app performs key management according to industry best practices

**Description**

This control covers the management of cryptographic keys throughout their lifecycle, including key generation, storage and protection. Crypto requirements only apply to sensitive data stored outside of the application sandbox.

**Rationale**

Even the strongest cryptography would be compromised by poor key management.

**Audit**

| Spec | Description |
|---|---|
| 1.2.2.1 | Cryptographic keys shall only be used for their defined purpose |
| 1.2.2.2 | Cryptographic key management shall be implemented properly |

## 1.3 Auth

### 1.3.1 The app uses secure authentication and authorization protocols and follows the relevant best practices

**Description**

Most apps connecting to a remote endpoint require user authentication and also enforce some kind of authorization. While the enforcement of these mechanisms must be on the remote endpoint, the apps also have to ensure that it follows all the relevant best practices to ensure a secure use of the involved protocols.

**Rationale**

Authentication and authorization provide an added layer of security and help prevent unauthorized access to sensitive user data.

**Audit**

| Spec | Description |
|---|---|
| 1.3.1.1 | If using OAuth 2.0 to authenticate, Proof Key for Code Exchange (PKCE) shall be implemented to protect the code grant |

## 1.4 Network

### 1.4.1 The app secures all network traffic according to the current best practices

**Description**

This control ensures that the app is in fact setting up secure connections in any situation. This is typically done by encrypting data and authenticating the remote endpoint, as TLS does. However, there are many ways for a developer to disable the platform secure defaults, or bypass them completely by using low-level APIs or third-party libraries.

**Rationale**

Ensuring data privacy and integrity of any data in transit is critical for any app that communicates over the network.

**Audit**

| Spec | Description |
|---|---|
| 1.4.1.1 | Network connections shall be encrypted |
| 1.4.1.2 | TLS configuration of network connections shall adhere to industry best practices |
| 1.4.1.3 | Endpoint identity shall be verified on network connections |

## 1.5 Platform

### 1.5.1 The app uses IPC mechanisms securely

**Description**

This control ensures that all interactions involving IPC mechanisms happen securely.

**Rationale**

Apps typically use platform provided IPC mechanisms to intentionally expose data or functionality. Both installed apps and the user are able to interact with the app in many different ways.

**Audit**

| Spec | Description |
|---|---|
| 1.5.1.1 | The app shall limit content provider exposure and harden queries against injection attacks |
| 1.5.1.2 | The app shall use verified links and sanitize all link input data |
| 1.5.1.3 | Any sensitive functionality exposed via IPC shall be intentional and at the minimum required level |
| 1.5.1.4 | All Pending Intents shall be immutable or otherwise justified for mutability |

### 1.5.2 The app uses WebViews securely

**Description**

This control ensures that WebViews are configured securely to prevent sensitive data leakage as well as sensitive functionality exposure (e.g. via JavaScript bridges to native code).

**Rationale**

WebViews are typically used by apps that have a need for increased control over the UI. They can, however, also be exploited by attackers or other installed apps, potentially compromising the app's security.

**Audit**

| Spec | Description |
|---|---|
| 1.5.2.1 | WebViews shall securely execute JavaScript |
| 1.5.2.2 | WebView shall be configured to allow the minimum set of protocol handlers required while disabling potentially dangerous handlers |

### 1.5.3 The app uses the user interface securely

**Description**

This control ensures that this data doesn't end up being unintentionally leaked due to platform mechanisms such as auto-generated screenshots or accidentally disclosed via e.g. shoulder surfing or sharing the device with another person.

**Rationale**

Sensitive data has to be displayed in the UI in many situations (e.g. passwords, credit card details, OTP codes in notifications) which can lead to unintentional leaks.

**Audit**

| Spec | Description |
|---|---|
| 1.5.3.1 | The app shall by default mask data in the User Interface when it is known to be sensitive |

## 1.6 Code

### 1.6.1 The app requires an up-to-date platform version

**Description**

This control ensures that the app is running on an up-to-date platform version so that users have the latest security protections.

**Rationale**

Every release of the mobile OS includes security patches and new security features. By supporting older versions, apps stay vulnerable to well-known threats.

**Audit**

| Spec | Description |
|---|---|
| 1.6.1.1 | The app shall set the targetSdkVersion to an up-to-date platform version |

### 1.6.2 The app only uses software components without known vulnerabilities

**Description**

To be truly secure, a full whitebox assessment should have been performed on all app components. However, as it usually happens with e.g. for third-party components this is not always feasible and not typically part of a penetration test. This control covers "low-hanging fruit" cases, such as those that can be detected just by scanning libraries for known vulnerabilities.

**Rationale**

The developer should protect users from known vulnerabilities.

**Audit**

| Spec | Description |
|---|---|
| 1.6.2.1 | The app only uses software components without known vulnerabilities |

### 1.6.3 The app validates and sanitizes all untrusted inputs

**Description**

Apps have many data entry points including the UI, IPC, the network, the file system, etc. This control ensures that this data is treated as untrusted input and is properly verified and sanitized before it's used.

**Rationale**

This incoming data might have been inadvertently modified by untrusted actors and may lead to bypass of critical security checks as well as classical injection attacks such as SQL injection, XSS or insecure deserialization.

**Audit**

| Spec | Description |
|---|---|
| 1.6.3.1 | Compiler security features shall be enabled |
| 1.6.3.2 | The App shall Mitigate Against Injection Flaws in Content Providers |
| 1.6.3.3 | Arbitrary URL redirects shall not be included in the app's webviews |
| 1.6.3.4 | Any use of implicit intents shall be appropriate for the app's functionality and any return data shall be handled securely |

## 1.7 Resilience

### 1.7.1 The app implements anti-tampering mechanisms

**Description**

This control tries to ensure the integrity of the app's intended functionality by preventing modifications to the original code and resources.

**Rationale**

Apps run on a user-controlled device, and without proper protections it's relatively easy to run a modified version locally (e.g. to cheat in a game, or enable premium features without paying), or upload a backdoored version of it to third-party app stores.

**Audit**

| Spec | Description |
|---|---|
| 1.7.1.1 | The app shall be properly signed |

### 1.7.2 The app implements anti-static analysis mechanisms

**Description**

This control tries to impede comprehension by making it as difficult as possible to figure out how an app works using static analysis.

**Rationale**

Understanding the internals of an app is typically the first step towards tampering with it.

**Audit**

| Spec | Description |
|---|---|
| 1.7.2.1 | The app shall disable all debugging symbols in the production version |

### 1.7.3 The app implements anti-dynamic analysis mechanisms

**Description**

Sometimes pure static analysis is very difficult and time consuming so it typically goes hand in hand with dynamic analysis. This control aims to make it as difficult as possible to perform dynamic analysis, as well as prevent dynamic instrumentation which could allow an attacker to modify the code at runtime.

**Rationale**

Observing and manipulating an app during runtime makes it much easier to decipher its behavior.

**Audit**

| Spec | Description |
|---|---|
| 1.7.3.1 | The app shall not be debuggable if installed from outside of commercial app stores |

## 1.8 Privacy

### 1.8.1 The app minimizes access to sensitive data and resources

**Description**

Apps should only request access to the data they absolutely need for their functionality and always with informed consent from the user. This control ensures that apps practice data minimization and restricts access control. Furthermore, apps should share data with third parties only when necessary, and this should include enforcing that third-party SDKs operate based on user consent, not by default or without it. Apps should prevent third-party SDKs from ignoring consent signals or from collecting data before consent is confirmed. Additionally, apps should be aware of the 'supply chain' of SDKs they incorporate, ensuring that no data is unnecessarily passed down their chain of dependencies.

**Rationale**

Data minimization reduces the potential impact of data breaches or leaks. This end-to-end responsibility for data aligns with recent SBOM regulatory requirements, making apps more accountable for their data practices.

**Audit**

| Spec | Description |
|---|---|
| 1.8.1.1 | The app shall minimize access to sensitive data and resources provided by the platform |

### 1.8.2 The app is transparent about data collection and usage

**Description**

This control ensures that apps provide clear information about data collection, storage, and sharing practices, including any behavior a user wouldn't reasonably expect, such as background data collection. Apps should also adhere to platform guidelines on data declarations.

**Rationale**

Users have the right to know how their data is being used.

**Audit**

| Spec | Description |
|---|---|
| 1.8.2.1 | The app shall be transparent about data collection and usage |

### 1.8.3 The app offers user control over their data

**Description**

This control ensures that apps provide mechanisms for users to manage, delete, and modify their data, and change privacy settings as needed (e.g. to revoke consent). Additionally, apps should re-prompt for consent and update their transparency disclosures when they require more data than initially specified.

**Rationale**

Users should have control over their data.

**Audit**

| Spec | Description |
|---|---|
| 1.8.3.1 | Users shall have the ability to request their data to be deleted via an in-app mechanism |

# 2 iOS

## 2.1 Storage

### 2.1.1 The app securely stores sensitive data in external storage

**Audit**

| Spec | Description |
|---|---|
| 2.1.1.1 | The app shall securely store sensitive data in external storage |

### 2.1.2 The app prevents leakage of sensitive data

**Audit**

| Spec | Description |
|---|---|
| 2.1.2.1 | The Keyboard Cache shall be Disabled for sensitive data inputs |
| 2.1.2.2 | No sensitive data shall be stored in system logs |

## 2.2 Crypto

### 2.2.1 The app employs current strong cryptography and uses it according to industry best practices

**Audit**

| Spec | Description |
|---|---|
| 2.2.1.1 | No insecure random number generators shall be utilized for any security sensitive context |
| 2.2.1.2 | Strong cryptography shall be implemented according to industry best practices |

### 2.2.2 The app performs key management according to industry best practices

**Audit**

| Spec | Description |
|---|---|
| 2.2.2.1 | Cryptographic keys shall only be used for their defined purpose |

| Spec | Description |
| --- | --- |
| 2.2.2.2 | Cryptographic key management shall be implemented properly |

## 2.3 Auth

### 2.3.1 The app uses secure authentication and authorization protocols and follows the relevant best practices

**Audit**

| Spec | Description |
| --- | --- |
| 2.3.1.1 | If using OAuth 2.0 to authenticate, Proof Key for Code Exchange (PKCE) shall be implemented to protect the code grant |

## 2.4 Network

### 2.4.1 The app secures all network traffic according to the current best practices

**Audit**

| Spec | Description |
| --- | --- |
| 2.4.1.1 | Network connections shall be encrypted |
| 2.4.1.2 | TLS configuration of network connections shall adhere to industry best practices |
| 2.4.1.3 | Endpoint identity shall be verified on network connections |

## 2.5 Platform

### 2.5.1 The app uses IPC mechanisms securely

**Audit**

| Spec | Description |
| --- | --- |
| 2.5.1.1 | The app shall not expose sensitive data via IPC mechanisms |
| 2.5.1.2 | The app shall not expose sensitive data via App Extensions |
| 2.5.1.3 | The app shall not expose sensitive functionality via Custom URL Schemes |
| 2.5.1.4 | The app shall not expose sensitive data via UIActivity Sharing |
| 2.5.1.5 | The app shall not use the general pasteboard for sharing sensitive information |
| 2.5.1.6 | The app shall not expose sensitive functionality via Universal Links |

### 2.5.2 The app uses WebViews securely

**Audit**

| Spec | Description |
|---|---|
| 2.5.2.1 | WebViews shall securely execute JavaScript |
| 2.5.2.2 | WebView shall be configured securely |

### 2.5.3 The app uses the user interface securely

**Audit**

| Spec | Description |
|---|---|
| 2.5.3.1 | The app shall by default mask data in the User Interface when it is known to be sensitive |

## 2.6 Code

### 2.6.1 The app requires an up-to-date platform version

**Audit**

| Spec | Description |
|---|---|
| 2.6.1.1 | The app shall set the targetSdkVersion to an up-to-date platform version |

### 2.6.2 The app only uses software components without known vulnerabilities

**Audit**

| Spec | Description |
|---|---|
| 2.6.2.1 | The app only uses software components without known vulnerabilities |

### 2.6.3 The app validates and sanitizes all untrusted inputs

**Audit**

| Spec | Description |
|---|---|
| 2.6.3.1 | Compiler security features shall be enabled |

## 2.7 Resilience

### 2.7.1 The app implements anti-tampering mechanisms

**Audit**

| Spec | Description |
|---|---|
| 2.7.1.1 | The app shall be properly signed |

### 2.7.2 The app implements anti-static analysis mechanisms

**Audit**

| Spec | Description |
|---|---|
| 2.7.2.1 | The app shall disable all debugging symbols in the production version |

### 2.7.3 The app implements anti-dynamic analysis techniques

**Audit**

| Spec | Description |
|---|---|
| 2.7.3.1 | The app shall not be debuggable if installed from outside of commercial app stores |

## 2.8 Privacy

### 2.8.1 The app minimizes access to sensitive data and resources

**Audit**

| Spec | Description |
|---|---|
| 2.8.1.1 | The app shall minimize access to sensitive data and resources provided by the platform |

### 2.8.2 The app is transparent about data collection and usage

**Audit**

| Spec | Description |
|---|---|
| 2.8.2.1 | The app shall be transparent about data collection and usage |

### 2.8.3 The app offers user control over their data

**Audit**

| Spec | Description |
|---|---|
| 2.8.3.1 | Users shall have the ability to request their data to be deleted via an in-app mechanism |

# App Defense Alliance Web Application Specification

Version 1.0 - 10-OCT 24

# Revision History

| Version | Date | Description |
|---|---|---|
| 0.5 | 5/25/24 | Initial draft based on Web App Tiger Team review of ASVS specification |
| 0.7 | 5/25/24 | Updates from Tiger Team review of 0.5 spec |
| 0.9 | 8/9/24 | Updates from ASA WG leads review of 0.7 spec |
| 1.0 | 10-OCT 24 | Approved for release by the alliance steering committee |

# Contributors

The App Defense Alliance Application Security Assessment Working Group (ASA WG) would like to thank the following individuals for their contributions to this specification.

## Application Security Assessment Working Group Leads

- Alex Duff (Meta) - ASA WG Chair
- Brooke Davis (Google) - ASA WG Vice Chair

## Web Profile Leads

- Brad Ree (Google)
- Michael Whiteman (Meta)

## Contributors

- Abdullah Albyati (Google)
- Alex Duff (Meta)
- Alexander Cobblah
- Anushree Shetty (KPMG)
- Artur Gartvikh
- Bhairavi Mehta (TAC Security)
- Brad Ree (Google)
- Brooke Davis (Google)
- Chilik Tamir
- Chris Cinnamo (Zimperium)
- Christopher Estrada (NCC Group)
- Cody Martin (Leviathan Security)
- Gianluca Braga (Zimperium)

- Joel Scambray (NCC Group)
- John Tidwell (Meta)
- Jorge Wallace Ruiz (Dekra)
- José María Santos López
- Juan Manuel Martinez Hernandez
- Julia McLaughlin (Google)
- Jullian Gerhart (NCC Group)
- Kelly Albrink (Bishop Fox)
- Mamachan Anish (KPMG)
- Manuel Mancera (Dekra)
- Mark Stribling (Leviathan Security)
- Mateo Morales Amador
- Michael Whiteman (Meta)
- Nazariy Haliley (Bishop Fox)
- Nico Chiaraviglio (Zimperium)
- Nicole Weisenbach (NCC Group)
- Noelle Murata (Leviathan Security)
- Pamela Dingle (Microsoft)
- Rene Guerra (Schellman)
- Richard Harris (NCC Group)
- Rupesh Nair (Net Sentries)
- Shad Malloy
- Soledad Antelada Toledano (Google)
- Tim Bolton (Meta)
- Viktor Sytnik (Leviathan)
- Zach Moreno (Bishop Fox)

# Table of Contents

# Introduction

In today's digitally-driven world, web applications are the backbone of countless businesses and organizations. Unfortunately, they are also prime targets for cyberattacks that threaten data confidentiality, service availability, and overall business integrity. To mitigate risks and build a secure web environment, a robust web application security standard and certification program is essential.

**Our Approach: OWASP ASVS as the Foundation**

This program leverages the internationally recognized OWASP Application Security Verification Standard (ASVS) as its core. The OWASP ASVS offers a comprehensive set of security assessment requirements and guidelines covering the entire web application development lifecycle. Building upon this base, the App Defense Alliance (ADA) focused on testable requirements with clear acceptance criteria. Further, the ADA approach emphasizes the use of automation where possible.

# Applicability

This document is intended for system and application administrators, security specialists, auditors, help desk, platform deployment, and/or DevOps personnel who plan to develop, deploy, assess, or secure solutions in

the cloud.

# References

1. OWASP Application Security Verification Standard

# Licensing



# Specification Scoping Guidance

Scoping a web application test can be a complex and challenging task, as it requires defining the boundaries of the application and determining what needs to be tested. Here is some guidance to help define a reasonable scope while evaluating a web application within the context of this specification.

This specification is designed to be applied to one or more target web application(s) (first party components) and their integrated third-party components, subject to the following considerations:

**First-Party Scoping Considerations**

- Target Web Application(s): Before testing, define the target web application(s) as a group of components that operate together to provide a logical set of services. For example, if a large business platform offers multiple services such as online dating, instant messaging, and investment banking; each can be scoped and evaluated separately using this specification, with all grouped subcomponents considered in-scope for testing & evaluation.
- Shared Backend Components: First party shared backend components or APIs are considered in the scope if they are utilized by the defined target application(s).

**Third-Party API Scoping Considerations**

- Sensitive Operations: Any third-party (3P) product API that supports sensitive operations, such as Authentication, accessing or mutating user data, & account recovery are within the scope of an ADA web assessment. These 3P APIs will only be subject to web assessment requirements defined within the Authentication, Session Management, & Access Control sections.
- Limited Testing: Testing of these 3P APIs will be limited to components and configurations utilized by the tested application. Other 3P API components and ADA web requirements will be out of scope and will not be tested in relation to 3P Product APIs.

**Additional Clarifications**

- Out-of-Scope Components: The following components are explicitly out of scope for this testing:
  - Third-party APIs not utilized by the target application
  - Non-sensitive operations performed by third-party APIs
- Examples and Scenarios: To illustrate the scoping decisions, consider the following examples:
  - A web application uses a third-party authentication API to handle user login. In this case, the authentication API is within scope for testing.

- A web application uses a third-party analytics gateway to process application performance metrics. As the gateway does not handle sensitive user data, it is out of scope for testing.

# Definitions

| Term | Definition |
|---|---|
| ADA-approved external user authentication service | User authentication / Identity provider which has been reviewed against the ADA authentication requirements. The review may have been done directly against the service or part of an application review. See ADA Approved User Authentication Service. |
| ASVS | Application Security Verification Standard |
| Authentication material | Sensitive information used to verify the identity of a user or service. These materials can include passwords, API tokens, session cookies, and other types of credentials that are used to authenticate access to a system or application. |
| Code snippets | Portion of code (either screenshot or text file), which demonstrates the implementation of the security control defined in the audit test case. The full source code, calling functions or underlying libraries do not need to be included. |
| Confidential data | Non-public information including user data and company confidential information which should only be accessible to authorized applications and systems. |
| CVE | Common Vulnerabilities and Exposures. https://www.cve.org/ |
| CVSS | Common Vulnerability Scoring System. https://www.cve.org/ |
| Default credentials | Default credentials are any predefined user names and passwords combinations. For example, Admin/Admin. However, Admin with a user defined password would not be a default credential. |
| HTTP parameter pollution | HTTP Parameter Pollution (HPP) is a web application vulnerability exploited by injecting encoded query string delimiters in already existing parameters. https://en.wikipedia.org/wiki/HTTP_parameter_pollution |
| IV (Initialization Vector) | A binary vector used as the input to initialize the algorithm for the encryption of a plaintext block sequence to increase security by introducing additional cryptographic variance and to synchronize cryptographic equipment. The initialization vector need not be secret. https://csrc.nist.gov/glossary/term/initialization_vector |
| Local File Inclusion | The File Inclusion vulnerability allows an attacker to include a file, usually exploiting a “dynamic file inclusion” mechanism implemented in the target application. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.1-Testing_for_Local_File_Inclusion |

| Term | Definition |
| --- | --- |
| (AL1) ADA Assurance Level 1 (Verified Self Assessment) | The developer provides evidence and statements of compliance to each audit test case. The ADA approved lab reviews the evidence against the requirements. The ADA approved lab does not directly assess the application. Some platform providers may require this testing and verification to be performed by an ASTL. |
| (AL2) ADA Assurance Level 2 (Lab Assessment) | The ADA approved lab evaluates each audit test case directly against the application. In some cases, the developer may need to provide limited information or code snippets. |
| Non-ADA approved authentication service | Any external user authentication service which has not been assessed against the ADA authentication requirements, or a developer's proprietary authentication service. |
| Padding oracle | A padding oracle is a function of an application which decrypts encrypted data provided by the client, e.g. internal session state stored on the client, and leaks the state of the validity of the padding after decryption. https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/09-Testing_for_Weak_Cryptography/02-Testing_for_Padding_Oracle |
| Principle of least privilege | A security principle that a system should restrict the access privileges of users (or processes acting on behalf of users) to the minimum necessary to accomplish assigned tasks. https://csrc.nist.gov/glossary/term/least_privilege |
| Publicly exposed interfaces | Any interface directly accessible on the Internet, either through a URL or IP address. Indirect access, such as access through a VPN or IP whitelisting, is out of scope. |
| Qualys SSL Labs scan | A free online service which performs a deep analysis of the configuration of any SSL/TLS web server on the public Internet. https://www.ssllabs.com/ssltest |
| Scope | Identifies whether a requirement is applicable to web applications, web APIs, or both. Mobile applications that utilize web APIs must comply with both the mobile application and web API specifications. |
| Remote File Inclusion | Remote File Inclusion (also known as RFI) is the process of including remote files through the exploitation of vulnerable inclusion procedures implemented in the application. https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/07-Input_Validation_Testing/11.2-Testing_for_Remote_File_Inclusion |
| WSTG | OWASP Web Security Testing Guide |
| 3P library | Any library which was not developed by the developer. These libraries may be open source or commercial libraries or SDKs. |

# 1 Authentication

## 1.1 Implement strong password security measures

### Description

Applications need to have robust mechanisms in place to ensure the security of user passwords. This includes, but is not limited to, enforcing password length requirements, implementing mitigations to prevent automated attacks against authentication systems, and securely storing passwords using strong cryptographic methods.

### Rationale

Weak or compromised passwords are a common attack vector used by adversaries to gain unauthorized access to user accounts. By implementing strong password security measures, organizations can significantly reduce the likelihood of successful password-based attacks.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 1.1.1 | Authentication is resistant to brute force attacks |
| 1.1.2 | System generated initial passwords or activation codes shall be securely randomly generated and expire after a short period. |
| 1.1.3 | Passwords shall be stored in a form that is resistant to offline attacks. |

## 1.2 Disable any default accounts for public application access interfaces

### Description

Applications should not have any pre-configured or default user accounts that can be used to access its public-facing interfaces. This includes both user and administrative accounts that come with default credentials.

### Rationale

Default accounts can be easily discovered through publicly available documentation, online forums, or other sources, making them an attractive target for attackers. If an attacker is able to gain access to a default account, they may be able to escalate their privileges and move laterally within the application or underlying infrastructure.

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 1.2.1 | Default credentials shall not present on publicly exposed interfaces. |

## 1.3 Out of band verifiers shall be random and not reused

### Description

Any verification codes or tokens sent through out-of-band methods (such as SMS or email) should have sufficient entropy along with a suitable expiration duration. Once a verifier has been used or has expired, it should be invalidated and a new one should be generated for each subsequent verification attempt.

### Rationale

By ensuring that out of band verifiers are securely generated and managed, the risk of an adversary intercepting and using these verifiers is significantly reduced.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 1.3.1 | Out of band verifier shall expire in a reasonable timeframe. |
| 1.3.2 | Out of band verifier shall only be used once. |
| 1.3.3 | Out of band verifier shall be securely random |
| 1.3.4 | Out of band verifier shall be resistant to brute force attacks |

# 2 Session Management

## 2.1 URLs shall not expose authentication material

### Description

Web applications must never expose authentication material, such as passwords or session cookies, within URL parameters. Authentication material should be transmitted securely, such as within HTTP headers or cookies with appropriate security flags.

### Rationale

Exposing authentication material such as session tokens in URLs significantly increases the risk of data loss and session hijacking. Attackers can easily intercept this data through browser history, network sniffing, or by

tricking users into visiting malicious links. This vulnerability undermines data protection, the security of user sessions and makes the application susceptible to unauthorized access

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 2.1.1 | The application shall not reveal passwords or session tokens in URL parameters. In cases where the application provides an API, the application shall prevent (or give developers an option) to prevent exposing sensitive information like API keys or session tokens within the URL query strings |

## 2.2 Implement session invalidation on logout, user request, and password change

### Description

The application must invalidate session tokens upon logout, expiration, and shall provide the option (or acts by default) to terminate other active sessions after a successful password change (including reset).

### Rationale

These features protect against unauthorized access. Logouts and expirations prevent lingering sessions, while password-change termination deters attackers who might know an old password. Session visibility and control let users proactively manage their account, ensuring that only authorized devices are actively associated with their profile.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 2.2.1 | Users shall have the ability to logout of the application. Logout or session expiration shall invalidate all stateful session tokens, including refresh tokens. |
| 2.2.2 | The application shall provide the option (or acts by default) to terminate all other active sessions, including stateful refresh tokens, after a successful password change (including change via password reset/recovery), and that this is effective across the application, federated login (if present), and any relying parties. |
| 2.2.3 | Non-revocable sateless authentication tokens must expire within 24 hours of being issued |

## 2.3 Implement and secure application session tokens

### Description

When using cookie-based session tokens, the application must enforce the 'Secure' attribute (ensuring transmission only over HTTPS) and the 'HttpOnly' attribute (preventing access by client-side JavaScript). The application prioritizes session tokens over static API keys, except where legacy systems necessitate static secrets.

### Rationale

Secure' and 'HttpOnly' mitigate risks of token interception and Cross-Site Scripting (XSS) attacks, enhancing session security. Session tokens, being temporary and user-specific, offer better control and auditing compared to long-lived API secrets, making them the preferred approach for modern applications.

### Scope

- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 2.3.1 | Cookie-based session tokens shall have the 'Secure' attribute set. |
| 2.3.2 | Cookie-based session tokens shall have the 'HttpOnly' attribute set. |
| 2.3.3 | The application shall use session tokens rather than static API secrets and keys, except with legacy implementations. |
| 2.3.4 | Stateless session tokens shall use digital signatures, encryption, and other countermeasures to protect against tampering, enveloping, replay, null cipher, and key substitution attacks. |

## 2.4 Protect sensitive account modifications

### Description

Applications must enforce a complete, valid login session or require re-authentication/secondary verification prior to any sensitive actions, such as sensitive data transactions or changes to account settings.

### Rationale

This requirement prevents unauthorized access to sensitive parts of an application. Even if an attacker partially compromises a session, re-authentication or secondary checks create an extra barrier. It helps mitigate session hijacking attempts and safeguards user data, promoting overall account security.

### Scope

- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|

| Spec | Description |
| --- | --- |
| 2.4.1 | Verify the application ensures a full, valid login session or requires re-authentication or secondary verification before allowing any sensitive transactions or account modifications. |

# 3 Access Control

## 3.1 Implement access control mechanisms to protect data and APIs

### Description

Applications shall enforce robust access controls at a trusted service layer, ensuring data integrity and applying the principle of least privilege. This includes protecting user/data attributes, limiting user manipulation, failing securely during exceptions, defending against Insecure Direct Object References (IDOR), and using strong anti-CSRF and multi-factor authentication (MFA) for administrative functions.

### Rationale

Enforcing least privilege access controls on a trusted service layer helps prevent unauthorized access and manipulation of sensitive data.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
| --- | --- |
| 3.1.1 | The application shall enforce least privilege access control rules on a trusted service layer. |
| 3.1.2 | All user and data attributes and policy information used by access controls shall not be able to be manipulated by end users unless specifically authorized. |
| 3.1.3 | Access controls shall fail securely including when an exception occurs. |
| 3.1.4 | Sensitive resources shall be protected against Insecure Direct Object Reference (IDOR) attacks targeting creation, reading, updating and deletion of records, such as creating or updating someone else's record, viewing everyone's records, or deleting all records. |
| 3.1.5 | The application or framework shall enforce a strong anti-CSRF mechanism to protect authenticated functionality, and effective anti-automation or anti-CSRF protects unauthenticated functionality. |
| 3.1.6 | Directory browsing shall be disabled unless deliberately desired. |

## 3.2 Implement secure OAuth integrations to protect user data and prevent unauthorized access

### Description

Applications which support OAuth integrations shall follow established security guidelines to safeguard user data and prevent unauthorized access.

### Rationale

OAuth is a widely adopted authorization framework that allows users to grant third-party applications limited access to their resources on another service without sharing their login credentials. However, if not implemented securely, OAuth can expose users to various attacks, including account compromises and information disclosure. By securely implementing OAuth integrations, the application minimizes these risks and provides users with a more secure experience.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 3.2.1 | Application shall implement only secure and recommended OAuth 2.0 flows, such as the Authorization Code Flow or the Authorization Code Flow with PKCE, while avoiding the use of deprecated flows like the Implicit Flow or the Resource Owner Password Credentials Flow. |
| 3.2.2 | Ensure that the application securely validates the redirect_uri and state parameters during the OAuth 2.0 authorization process to prevent open redirect and CSRF vulnerabilities. |

## 3.3 Application exposed administrative interfaces shall use appropriate multi-factor authentication.

### Description

Application exposed administrative interfaces shall implement multi-factor authentication. These interfaces shall be limited to application layer functionality and must not expose the cloud infrastructure.

### Rationale

Infrastructure administrative interfaces shall never be exposed through an internet facing interface. However, there are many cases where application layer administrative tasks may need to be exposed to the internet. It is critical that these interfaces be limited in functionality and always implement multi-factor authentication to prevent attackers from compromising administrative accounts.

### Scope

- Web application

### Audit

| Spec | Description |
| --- | --- |
| 3.3.1 | Application administrative interfaces shall use appropriate multi-factor authentication to prevent unauthorized use. |

# 4 Communications

## 4.1 Protect data through strong cryptography

### Description

Applications must enforce strong TLS configurations and cryptographic practices. This includes using up-to-date tools to enable only strong cipher suites (prioritizing the strongest), employing trusted TLS certificates, and ensuring secure failure modes in cryptographic modules to mitigate common cryptographic attacks.

### Rationale

Strong TLS and cipher suites ensure confidentiality and integrity of data in transit by protecting against eavesdropping and modification. Trusted TLS certificates verify authenticity and prevent adversary-in-the-middle attacks, while secure failure modes and robust cryptography deter advanced attacks exploiting weaknesses in cryptographic implementations.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
| --- | --- |
| 4.1.1 | Application shall enforce the use of TLS for all connections and default to TLS 1.2+. In cases where support for legacy clients is necessary, TLS 1.0 and 1.1 may be supported if mitigations are implemented to minimize the risk of downgrade attacks and known TLS exploits. Regardless of the TLS version in use, the application shall default to secure cipher suites and reject those with known vulnerabilities. |
| 4.1.2 | Connections to and from the server shall use trusted TLS certificates. Where internally generated or self-signed certificates are used, the server must be configured to only trust specific internal CAs and specific self-signed certificates. All others should be rejected. |
| 4.1.3 | No instances of weak cryptography which meaningfully impact the confidentiality or integrity of confidential data. |
| 4.1.4 | All cryptographic modules shall fail securely, and errors are handled in a way that does not enable Padding Oracle attacks. |

# 5 Data Validation and Sanitization

## 5.1 Implement validation & input sanitation

### Description

Web applications must implement robust input validation and output encoding to defend against a wide range of injection attacks. This includes protecting against HTTP Parameter Pollution, XSS (reflected, stored, and DOM-based), SQL injection, OS command injection, file inclusion vulnerabilities, template injection, SSRF, XPath/XML injection, and unsafe use of dynamic code execution features (like eval()).

### Rationale

Robust input validation and output encoding is essential for web applications to effectively defend against multiple injection attack types. Injection attacks pose a significant risk for web applications due to their simplicity and ease of automation, enabling potential attackers to readily target vulnerable sites. By implementing secure input validation, web applications can significantly reduce the risk of attackers exploiting injection vulnerabilities to gain unauthorized access, manipulate data, or compromise systems.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 5.1.1 | Protect against HTTP parameter pollution. |
| 5.1.2 | URL redirects and forwards are limited to allowlisted URLs or a warning is displayed when redirecting to untrusted content. |
| 5.1.3 | Avoid the use of eval() or other dynamic code execution features. When there is no alternative, any user input is sanitized and sandboxed before being executed. |
| 5.1.4 | Protect against template injection attacks by ensuring that any user input being included is sanitized or sandboxed. |
| 5.1.5 | Prevent Server-Side Request Forgery (SSRF) |
| 5.1.6 | Protect against XPath or XML injection attacks |
| 5.1.7 | Context-aware output escaping or sanitization protects against reflected, stored, and DOM based XSS. |
| 5.1.8 | Protect against database injection attacks |
| 5.1.9 | Protect against OS command injections |
| 5.1.10 | Protect against local file inclusion or remote file inclusion attacks |

## 5.2 Securely Handle Untrusted Files

### Description

Web applications must safely process and manage files that originate from untrusted or unknown sources. This includes restricting uploads to expected file types and preventing direct execution of uploaded content containing HTML, JavaScript, or dynamic server-side code.

### Rationale

Files from untrusted sources may contain malicious code which could allow compromise of the application. If these files are executed directly, they can compromise the security of the web application, leading to unauthorized access, data breaches, or other harmful actions.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 5.2.1 | Protect against malicious file uploads by limiting uploads to expected file types and preventing direct execution of uploaded content. |

# 6 Configuration

## 6.1 Keep all components up to date

### Description

Developers must verify that the libraries included in their application do not have any known exploitable vulnerabilities.

### Rationale

Attackers can perform automated scans to identify vulnerable applications based on published vulnerabilities.

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 6.1.1 | The app only uses software components without known exploitable vulnerabilities. |

## 6.2 Disable debug modes in production environments

### Description

Applications must strictly disable all debug modes before deployment into production environments.

### Rationale

Debug modes often expose sensitive information like stack traces, code internals, and environment variables. This information can aid attackers in understanding the application's structure and identifying vulnerabilities, significantly increasing the risk of targeted attacks and exploitation. Disabling debug modes removes this unnecessary risk in production.

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 6.2.1 | Disable debug modes in production environments |

## 6.3 The origin header shall not be used for authentication of access control decisions

### Description

The application must never rely solely on the Origin HTTP header for authentication or access control decisions.

### Rationale

The Origin header can be easily manipulated by attackers, making it an unreliable indicator of a request's true source. This could lead to unauthorized access if an application mistakenly trusts requests based on a forged Origin header. Security mechanisms must use more robust and tamper-proof methods for authentication and authorization.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 6.3.1 | The origin header shall not be used for authentication of access control decisions |

## 6.4 Protect Application from Subdomain Takeover

### Description

The application must implement safeguards to prevent subdomain takeover vulnerabilities. This includes proactive identification and removal of dangling DNS records (e.g., CNAME records pointing to decommissioned services) and regular monitoring of third-party services integrated with the application's domains.

### Rationale

Dangling DNS records and vulnerable third-party services can allow attackers to take control of subdomains. This could enable them to host malicious content on the application's domain, harming reputation and potentially leading to phishing attacks or the compromise of user data.

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 6.4.1 | The application shall not be susceptible to subdomain takeovers. |

## 6.5 Do not log credentials or payment details

### Description

Applications must never log authentication material, such as user credentials (e.g., passwords, API keys) or payment details (e.g., credit card numbers, CVVs).

### Rationale

Many data privacy regulations (PCI-DSS, GDPR, etc.) explicitly prohibit the storage of sensitive authentication and financial data, especially in plaintext. In addition, avoiding logging sensitive information minimizes the overall attack surface and demonstrates a commitment to responsible data handling.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 6.5.1 | The application shall not log credentials or payment details. Session tokens shall only be stored in logs in an irreversible, hashed form. |

## 6.6 Securely clear client storage during logout

### Description

Web applications should ensure that any confidential data or authentication material stored in the browser's local storage is deleted or otherwise rendered inaccessible when the user logs out.

### Rationale

Properly deleting confidential data and authentication material after logout decreases the risk that an attacker with local access to the system will be able to compromise the data. This is particularly relevant in scenarios where users are logging in from shared systems or devices.

### Scope

- Web application

### Audit

| Spec | Description |
|---|---|
| 6.6.1 | Browser storage is securely cleared during logout. |

## 6.7 Securely store server-side secrets

### Description

Ensure server-side secrets are stored securely using an appropriate secrets management approach which provides encryption, access controls, and monitoring to prevent unauthorized access and maintain data confidentiality.

### Rationale

Secrets management helps protect API keys, access tokens, and other server-side secrets used by the application from being accessed or stolen by unauthorized parties.

### Scope

- Web application
- Web and mobile APIs

### Audit

| Spec | Description |
|---|---|
| 6.7.1 | The application shall securely store access tokens, API keys, and other server-side secrets. |

# App Defense Alliance Cloud Profile

## Specification

Version 1.0 - 10-OCT 24

# Revision History

| Version | Date | Description |
|---|---|---|
| 0.5 | 8-MAY 24 | Initial draft based on Web App Tiger Team review of CIS Cloud Foundations Benchmarks |
| 0.7 | 25-MAY 24 | Updates from Tiger Team review of 0.5 spec |
| 0.9 | 9-AUG 24 | Updates from ASA WG leads review of 0.7 spec |
| 1.0 | 10-OCT 24 | Approved for release by the alliance steering committee |

# Contributors

The App Defense Alliance Application Security Assessment Working Group (ASA WG) would like to thank the following individuals for their contributions to this specification.

### Application Security Assessment Working Group Leads

- Alex Duff (Meta) - ASA WG Chair
- Brooke Davis (Google) - ASA WG Vice Chair

### Cloud Profile Leads

- Alex Duff (Meta)
- Brad Ree (Google)

### Contributors

- Alex Duff (Meta)
- Andrew Kiggins
- Anushree Shetty (KPMG)
- Artur Gartvikh
- Bhairavi Mehta (TAC Security)
- Brad Ree (Google)
- Chris Schneider (Zimperium)
- Christopher Estrada (NCC Group)
- Cody Martin (Leviathan Security)

- Gianluca Braga (Zimperium)
- Joel Scambray (NCC Group)
- Jorge Wallace (Dekra)
- José María Santos López
- Juan Manuel Martinez Hernandez
- Julia McLaughlin (Google)
- Mamachan Anish (KPMG)
- Mark Stribling (Leviathan Security)
- Mateo Morales Amador
- Michael Whiteman (Meta)
- Nazariy Haliley (Bishop Fox)
- noelle murata (Leviathan Security)
- Pamela Dingle (Microsoft)
- Rene Guerra (Schellman)
- Rennie deGraaf (NCC Group)
- Richard Harris (NCC Group)
- Rupesh Nair (Net Sentries)
- Ryan Nakamoto (Meta)
- Sebastian Porst
- Shad Malloy
- Tim Bolton (Meta)
- Tony Balkan (Microsoft)
- Viktor Sytnik (Leviathan Security)

# Table of Contents

6.15 Collect Detailed Audit Logs

# Overview

This document provides prescriptive guidance for configuring security options for a subset of cloud services offered by Amazon Web Services, Google Cloud Platform, and Microsoft Azure. This profile emphasizes foundational, testable, and architecture agnostic settings that are suitable for applications that process non-public data such as user data, user device data, company data, credentials, keys, or other types of confidential information. Note that apps in certain verticals such as healthcare or finance may have to meet higher security, privacy, and regulatory requirements.

# Applicability

This document is intended for system and application administrators, security specialists, auditors, help desk, platform deployment, and/or DevOps personnel who plan to develop, deploy, assess, or secure solutions in the cloud.

# Acknowledgements

This profile builds upon the work of the Center for Internet Security (CIS), specifically their cloud foundations benchmarks.

1. CIS Amazon Web Services Foundations Benchmark v2.0.0
2. CIS Google Cloud Platform Foundation Benchmark v2.0.0
3. CIS Microsoft Azure Foundations Benchmark v2.0.0

# Licensing



# 1 Compute

## 1.1 Establish and Maintain a Software Inventory

### Description

Establish and maintain a detailed inventory of all licensed software installed on enterprise assets. The software inventory must document the title, publisher, initial install/use date, and business purpose for each entry; where appropriate, include the Uniform Resource Locator (URL), app store(s), version(s), deployment mechanism, and decommission date. Review and update the software inventory bi-annually, or more frequently.

### Rationale

It is necessary to first identify the software that needs to be secured before taking additional steps towards achieving a suitable security baseline.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.1.1 | Azure | Ensure that Only Approved Extensions Are Installed |

## 1.2 Ensure Authorized Software is Currently Supported

### Description

Ensure that only currently supported software is designated as authorized in the software inventory for enterprise assets. If software is unsupported, yet necessary for the fulfillment of the enterprise's mission, document an exception detailing mitigating controls and residual risk acceptance. For any unsupported software without an exception documentation, designate as unauthorized. Review the software list to verify software support at least monthly, or more frequently.

### Rationale

When software ceases to be supported, the maintainer of that software will no longer issue patches to remediate security vulnerabilities that are discovered in it. This leaves any organization relying on that software at a high risk of a security incident.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.2.1 | AWS | Ensure that all AWS Lambda functions are configured to use a current (not deprecated) runtime |
| 1.2.3 | Azure | Ensure That 'PHP version' is the Latest, If Used to Run the Web App |
| 1.2.4 | Azure | Ensure that 'Python version' is the Latest Stable Version, if Used to Run the Web App |
| 1.2.5 | Azure | Ensure that 'Java version' is the latest, if used to run the Web App |
| 1.2.6 | Azure | Ensure that 'HTTP Version' is the Latest, if Used to Run the Web App |
| 1.2.6 | Google | Ensure that all GCP Cloud functions are configured to use a current (not deprecated) runtime |

## 1.3 Encrypt Confidential Data in Transit

### Description

Encrypt confidential data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).

### Rationale

Encryption protects confidential data when transmitted over untrusted network connections.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.3.1 | Azure | Ensure Web App Redirects All HTTP traffic to HTTPS in Azure App Service |
| 1.3.2 | Azure | Ensure Web App is using the latest version of TLS encryption |
| 1.3.3 | Azure | Ensure FTP deployments are Disabled |
| 1.3.4 | Google | Ensure “Block Project-Wide SSH Keys” Is Enabled for VM Instances |

## 1.4 Encrypt Confidential Data at Rest

### Description

Encrypt confidential data at rest on servers, applications, and databases. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.

### Rationale

Encryption at rest protects against some risks of unauthorized access to data, for example insecure disposal and reuse of storage media.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.4.1 | Azure | Ensure Virtual Machines are utilizing Managed Disks |

## 1.5 Implement and Manage a Firewall on Servers

### Description

Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.

### Rationale

Firewalls help to prevent unauthorized users from accessing servers or sending malicious payloads to those servers.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.5.1 | Google | Ensure That IP Forwarding Is Not Enabled on Instances |

## 1.6 Manage Default Accounts on Enterprise Assets and Software

### Description

Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.

### Rationale

Products typically ship with insecure defaults that, if not configured securely, can be used by malicious users to take over a system.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.6.1 | Google | Ensure That Instances Are Not Configured To Use the Default Service Account |
| 1.6.2 | Google | Ensure That Instances Are Not Configured To Use the Default Service Account With Full Access to All Cloud APIs |

## 1.7 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software

### Description

Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.

### Rationale

Uninstalling and disabling unnecessary services reduces the target area of your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.7.1 | Google | Ensure ‘Enable Connecting to Serial Ports’ Is Not Enabled for VM Instance |

## 1.8 Centralize Account Management

### Description

Centralize account management through a directory or identity service.

### Rationale

Centralizing makes administration simpler and therefore reduces risks related to unauthorized account creation or usage.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 1.8.1 | Azure | Ensure that Register with Azure Active Directory is enabled on App Service |
| 1.8.2 | Google | Ensure Oslogin Is Enabled for a Project |

# 2 Identity and Access Management

## 2.1 Establish and Maintain a Data Recovery Process

### Description

Establish and maintain a data recovery process. In the process, address the scope of data recovery activities, recovery prioritization, and the security of backup data. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

"Organizations need to establish and maintain data recovery practices sufficient to restore in-scope enterprise assets to a pre-incident and trusted state."

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.1.1 | Azure | Ensure the Key Vault is Recoverable |

## 2.2 Designate Personnel to Manage Incident Handling

### Description

Designate one key person, and at least one backup, who will manage the enterprise’s incident handling process. Management personnel are responsible for the coordination and documentation of incident response and recovery efforts and can consist of employees internal to the enterprise, third-party vendors, or a hybrid approach. If using a third-party vendor, designate at least one person internal to the enterprise to oversee any third-party work. Review annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

Without an incident response plan, an enterprise may not discover an attack in the first place, or, if the attack is detected, the enterprise may not follow good procedures to contain damage, eradicate the attacker’s presence, and recover in a secure fashion.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.2.1 | AWS | Ensure a support role has been created to manage incidents with AWS Support |

## 2.3 Establish and Maintain Contact Information for Reporting Security Incidents

### Description

Establish and maintain contact information for parties that need to be informed of security incidents. Contacts may include internal staff, third-party vendors, law enforcement, cyber insurance providers, relevant government agencies, Information Sharing and Analysis Center (ISAC) partners, or other stakeholders. Verify contacts annually to ensure that information is up-to-date.

### Rationale

As time goes by -- and processes and people change within an organization -- it's important to keep contact information up to date so that information about a security incident reaches the right individuals promptly.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.3.1 | AWS | Maintain current contact details |
| 2.3.2 | AWS | Ensure security contact information is registered |
| 2.3.5 | Google | Ensure Essential Contacts is Configured for Organization |

## 2.4 Address Unauthorized Software

### Description

Ensure that unauthorized software is either removed from use on enterprise assets or receives a documented exception. Review monthly, or more frequently.

### Rationale

Actively manage (inventory, track, and correct) all software (operating systems and applications) on the network so that only authorized software is installed and can execute, and that unauthorized and unmanaged software is found and prevented from installation or execution.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.4.1 | Azure | Ensure `User consent for applications` is set to `Do not allow user consent` |
| 2.4.2 | Azure | Ensure that 'Users can add gallery apps to My Apps' is set to 'No' |

| Spec | Platform | Description |
|---|---|---|
| 2.4.3 | Azure | Ensure That 'Users Can Register Applications' Is Set to 'No' |

## 2.5 Establish and Maintain a Data Management Process

### Description

Establish and maintain a data management process. In the process, address data sensitivity, data owner, handling of data, data retention limits, and disposal requirements, based on sensitivity and retention standards for the enterprise. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

Develop processes and technical controls to identify, classify, securely handle, retain, and dispose of data.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.5.1 | Azure | Ensure that the Expiration Date that is no more than 90 days in the future is set for all Keys in RBAC Key Vaults |
| 2.5.2 | Azure | Ensure that the Expiration Date that is no more than 90 days in the future is set for all Keys in Non-RBAC Key Vaults. |
| 2.5.3 | Azure | Ensure that the Expiration Date that is no more than 90 days in the future is set for all Secrets in RBAC Key Vaults |
| 2.5.4 | Azure | Ensure that the Expiration Date that is no more than 90 days in the future is set for all Secrets in Non-RBAC Key Vaults |

## 2.6 Encrypt Confidential Data at Rest

### Description

Encrypt confidential data at rest on servers, applications, and databases. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.

### Rationale

Encryption at rest protects against some risks of unauthorized access to data, for example insecure disposal and reuse of storage media.

### Audit

| Spec | Platform | Description |
|---|---|---|

| Spec | Platform | Description |
|---|---|---|
| 2.6.1 | Google | Ensure Secrets are Not Stored in Cloud Functions Environment Variables by Using Secret Manager |

## 2.7 Configure Data Access Control Lists

### Description

Configure data access control lists based on a user’s need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.

### Rationale

The principle of least privilege reduces the risk of unauthorized actions being taken in your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.7.1 | AWS | Ensure no 'root' user account access key exists |
| 2.7.2 | Aws | Do not setup access keys during initial user setup for all IAM users that have a console password |
| 2.7.3 | AWS | Ensure IAM policies that allow full ":" administrative privileges are not attached |
| 2.7.4 | Azure | Ensure That 'Guest users access restrictions' is set to 'Guest user access is restricted to properties and memberships of their own directory objects' |
| 2.7.5 | Google | Ensure That IAM Users Are Not Assigned the Service Account User or Service Account Token Creator Roles at Project Level |
| 2.7.6 | Google | Ensure That Cloud KMS Cryptokeys Are Not Anonymously or Publicly Accessible |

## 2.8 Establish and Maintain a Secure Configuration Process

### Description

Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

"""This CIS Control provides guidance for securing hardware and software. As delivered by the CSP, the default configurations for operating systems and applications are normally geared toward ease-of-deployment and ease-of-use -- not security. Basic controls, open services and ports, default accounts or passwords, older (vulnerable) protocols, pre-installation of unneeded software -- all can be exploitable in their default state. Even if a strong initial configuration is developed and deployed in the cloud, it must be

continually managed to avoid configuration drift as software is updated or patched, new security vulnerabilities are reported, and configurations are “tweaked” to allow the installation of new software or to support new operational requirements. If not, attackers will find opportunities to exploit both network-accessible services and client software."""

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.8.1 | Azure | Ensure Security Defaults is enabled on Azure Active Directory |
| 2.8.2 | AWS | Ensure IAM password policy requires minimum length of 14 or greater |
| 2.8.3 | AWS | Ensure there is only one active access key available for any single IAM user |
| 2.8.4 | AWS | Ensure access keys are rotated every 90 days or less |

## 2.9 Use Unique Passwords

### Description

Use unique passwords for all enterprise assets. Best practice implementation includes, at a minimum, an 8-character password for accounts using MFA and a 14-character password for accounts not using MFA.

### Rationale

Malicious users automate login attempts using username and password databases from breaches of other systems. Password policies can help to reduce the risk of a breached or otherwise insecure password being used.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.9.1 | AWS | Ensure IAM password policy prevents password reuse |
| 2.9.2 | Azure | Ensure that a Custom Bad Password List is set to 'Enforce' for your Organization |

## 2.10 Disable Dormant Accounts

### Description

Delete or disable any dormant accounts after a period of 45 days of inactivity, where supported.

### Rationale

Ensuring that dormant accounts are disabled when they're no longer needed reduces the target area for malicious users.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.10.1 | AWS | Ensure credentials unused for 45 days or greater are disabled |
| 2.10.2 | Azure | Ensure Guest Users Are Reviewed on a Regular Basis |

## 2.11 Restrict Administrator Privileges to Dedicated Administrator Accounts

### Description

Restrict administrator privileges to dedicated administrator accounts on enterprise assets. Conduct general computing activities, such as internet browsing, email, and productivity suite use, from the user’s primary, non-privileged account.

### Rationale

As a matter of good practice, users who can take administrative actions should use regular permissions for routine actions that do not require administrative privileges. This reduces the damage that could occur if the user encounters a malicious exploit attempt.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.11.1 | AWS | Eliminate use of the 'root' user for administrative and daily tasks |
| 2.11.2 | Azure | Ensure That 'Notify all admins when other admins reset their password?' is set to 'Yes' |
| 2.11.3 | Azure | Ensure That 'Restrict access to Azure AD administration portal' is Set to 'Yes' |
| 2.11.4 | Azure | Ensure That No Custom Subscription Administrator Roles Exist |
| 2.11.5 | Google | Ensure That Service Account Has No Admin Privileges |

## 2.12 Centralize Account Management

### Description

Centralize account management through a directory or identity service.

### Rationale

Centralizing makes administration simpler and therefore reduces risks related to unauthorized account creation or usage.

### Audit

| Spec | Platform | Description |
|---|---|---|

| Spec | Platform | Description |
|---|---|---|
| 2.12.1 | Google | Ensure that Corporate Login Credentials are Used |

## 2.13 Establish an Access Revoking Process

### Description

Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.

### Rationale

Ensuring that access grants are revoked when they're no longer needed reduces the target area for malicious users.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.13.1 | Azure | Ensure that 'Number of days before users are asked to re-confirm their authentication information' is set to '90' |

## 2.14 Require MFA for Externally-Exposed Applications

### Description

Require all externally-exposed enterprise or third-party applications to enforce MFA, where supported. Enforcing MFA through a directory service or SSO provider is a satisfactory implementation of this Safeguard.

### Rationale

Requiring MFA makes it harder for malicious attackers to takeover accounts, e.g., by re-using username and password combinations that have become leaked from other systems

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.14.1 | Azure | Ensure That 'Number of methods required to reset' is set to '2' |
| 2.14.2 | Azure | Ensure that 'Require Multi-Factor Authentication to register or join devices with Azure AD' is set to 'Yes' |
| 2.14.3 | Azure | Ensure that 'Multi-Factor Auth Status' is 'Enabled' for all Privileged Users |
| 2.14.4 | Azure | Ensure that 'Allow users to remember multi-factor authentication on devices they trust' is Disabled |
| 2.14.5 | Azure | Ensure that A Multi-factor Authentication Policy Exists for All Users |

| Spec | Platform | Description |
|---|---|---|
| 2.14.6 | Azure | Ensure Multi-factor Authentication is Required for Risky Sign-ins |
| 2.14.7 | Google | Ensure that Multi-Factor Authentication is 'Enabled' for All Non-Service Accounts |
| 2.14.8 | Azure | Ensure that 'Multi-Factor Auth Status' is 'Enabled' for all Non-Privileged Users |

## 2.15 Require MFA for Remote Network Access

### Description

Require MFA for remote network access.

### Rationale

Requiring MFA makes it harder for malicious attackers to takeover accounts, e.g., by re-using username and password combinations that have become leaked from other systems

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.15.1 | Azure | Ensure that A Multi-factor Authentication Policy Exists for Administrative Groups |
| 2.15.2 | Azure | Ensure Multi-factor Authentication is Required for Azure Management |

## 2.16 Require MFA for Administrative Access

### Description

Require MFA for all administrative access accounts, where supported, on all enterprise assets, whether managed on-site or through a third-party provider.

### Rationale

Requiring MFA makes it harder for malicious attackers to takeover accounts, e.g., by re-using username and password combinations that have become leaked from other systems

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.16.1 | AWS | Ensure MFA is enabled for the 'root' user account |

## 2.17 Centralize Access Control

### Description

Centralize access control for all enterprise assets through a directory service or SSO provider, where supported.

### Rationale

Centralizing makes administration simpler and therefore reduces risks related to unauthorized account creation or usage.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.17.1 | Azure | Ensure that 'Notify users on password resets?' is set to 'Yes' |

## 2.18 Define and Maintain Role-Based Access Control

### Description

Define and maintain role-based access control, through determining and documenting the access rights necessary for each role within the enterprise to successfully carry out its assigned duties. Perform access control reviews of enterprise assets to validate that all privileges are authorized, on a recurring schedule at a minimum annually, or more frequently.

### Rationale

Standardizing the mechanism for granting cloud permissions reduces the risk of an unintentional or unnoticed privilege.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 2.18.1 | AWS | Ensure IAM Users Receive Permissions Only Through Groups |

# 3 Logging and Monitoring

## 3.1 Establish and Maintain Detailed Enterprise Asset Inventory

### Description

Establish and maintain an accurate, detailed, and up-to-date inventory of all enterprise assets with the potential to store or process data, to include: end-user devices (including portable and mobile), network devices, non-computing/IoT devices, and servers. Ensure the inventory records the network address (if static), hardware address, machine name, enterprise asset owner, department for each asset, and whether the asset has been approved to connect to the network. For mobile end-user devices, MDM type tools can support this process, where appropriate. This inventory includes assets connected to the infrastructure physically, virtually, remotely, and those within cloud environments. Additionally, it includes assets that are regularly connected to

the enterprise’s network infrastructure, even if they are not under control of the enterprise. Review and update the inventory of all enterprise assets bi-annually, or more frequently.

### Rationale

It is necessary to first identify the systems and devices that need to be secured before taking additional steps towards achieving a suitable security baseline.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.1.1 | Google | Ensure Cloud Asset Inventory Is Enabled |

## 3.2 Tune Security Event Alerting Thresholds

### Description

Tune security event alerting thresholds monthly, or more frequently.

### Rationale

Tools must be tuned to reduce the prevalence of both false negatives and false positives.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.2.1 | Azure | Ensure That 'Notify about alerts with the following severity' is Set to 'High' |

## 3.3 Establish and Maintain Contact Information for Reporting Security Incidents

### Description

Establish and maintain contact information for parties that need to be informed of security incidents. Contacts may include internal staff, third-party vendors, law enforcement, cyber insurance providers, relevant government agencies, Information Sharing and Analysis Center (ISAC) partners, or other stakeholders. Verify contacts annually to ensure that information is up-to-date.

### Rationale

As time goes by -- and processes and people change within an organization -- it's important to keep contact information up to date so that information about a security incident reaches the right individuals promptly.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.3.1 | Azure | Ensure That 'All users with the following roles' is set to 'Owner' |

| Spec | Platform | Description |
|---|---|---|
| 3.3.2 | Azure | Ensure 'Additional email addresses' is Configured with a Security Contact Email |

## 3.4 Log Confidential Data Access

### Description

Log confidential data access, including modification and disposal.

### Rationale

Organizations need reliable forensic information about access, modification, and deletion of confidential data.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.4.1 | AWS | Ensure S3 bucket access logging is enabled on the CloudTrail S3 bucket |

## 3.5 Configure Data Access Control Lists

### Description

Configure data access control lists based on a user’s need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.

### Rationale

The principle of least privilege reduces the risk of unauthorized actions being taken in your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.5.1 | AWS | Ensure the S3 bucket used to store CloudTrail logs is not publicly accessible |
| 3.5.2 | Azure | Ensure the Storage Container Storing the Activity Logs is not Publicly Accessible |

## 3.6 Establish and Maintain a Secure Configuration Process

### Description

Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

This CIS Control provides guidance for securing hardware and software. As delivered by the CSP, the default configurations for operating systems and applications are normally geared toward ease-of-deployment and ease-of-use -- not security. Basic controls, open services and ports, default accounts or passwords, older (vulnerable) protocols, pre-installation of unneeded software -- all can be exploitable in their default state. Even if a strong initial configuration is developed and deployed in the cloud, it must be continually managed to avoid configuration drift as software is updated or patched, new security vulnerabilities are reported, and configurations are “tweaked” to allow the installation of new software or to support new operational requirements. If not, attackers will find opportunities to exploit both network- accessible services and client software.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.6.1 | Azure | Ensure Any of the ASC Default Policy Settings are Not Set to 'Disabled' |

## 3.7 Perform Automated Operating System Patch Management

### Description

Perform operating system updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.

### Rationale

Patching remediates known vulnerabilities. Using automation makes this process routine and reduces the window of opportunity for attackers.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.7.1 | Azure | Ensure that Microsoft Defender Recommendation for 'Apply system updates' status is 'Completed' |

## 3.8 Perform Automated Vulnerability Scans of Internal Enterprise Assets

### Description

Perform automated vulnerability scans of internal enterprise assets on a quarterly, or more frequent, basis. Conduct both authenticated and unauthenticated scans, using a SCAP-compliant vulnerability scanning tool.

### Rationale

Tools can help to identify vulnerabilities that require remediation.

### Audit

| Spec | Platform | Description |
|---|---|---|

| Spec | Platform | Description |
|---|---|---|
| 3.8.1 | Azure | Ensure that Auto provisioning of 'Log Analytics agent for Azure VMs' is Set to 'On' |

## 3.9 Conduct Audit Log Reviews

### Description

Conduct reviews of audit logs to detect anomalies or abnormal events that could indicate a potential threat. Conduct reviews on a weekly, or more frequent, basis.

### Rationale

Logs may contain indications of compromise, so it's important to review logs regularly to detect and stop unauthorized or destructive actions from taking place in your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.9.1 | AWS | Ensure management console sign-in without MFA is monitored |
| 3.9.2 | AWS | Ensure usage of 'root' account is monitored |
| 3.9.3 | AWS | Ensure IAM policy changes are monitored |
| 3.9.4 | AWS | Ensure CloudTrail configuration changes are monitored |
| 3.9.5 | AWS | Ensure S3 bucket policy changes are monitored |
| 3.9.6 | AWS | Ensure changes to network gateways are monitored |
| 3.9.7 | AWS | Ensure route table changes are monitored |
| 3.9.8 | AWS | Ensure VPC changes are monitored |
| 3.9.9 | AWS | Ensure AWS Organizations changes are monitored |
| 3.9.10 | Google | Ensure That Cloud Audit Logging Is Configured Properly |
| 3.9.11 | Google | Ensure That Cloud DNS Logging Is Enabled for All VPC Networks |

## 3.10 Collect Audit Logs

### Description

Collect audit logs. Ensure that logging, per the enterprise’s audit log management process, has been enabled across enterprise assets and that logs are retained for at least a minimum period of time.

### Rationale

Having log files of what actions have taken place by users and also system events is fundamental to being able to detect security events.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.10.1 | Google | Ensure That Sinks Are Configured for All Log Entries |
| 3.10.2 | Google | Ensure Log Metric Filter and Alerts Exist for Project Ownership Assignments/Changes |
| 3.10.3 | Google | Ensure That the Log Metric Filter and Alerts Exist for Audit Configuration Changes |
| 3.10.4 | Google | Ensure That the Log Metric Filter and Alerts Exist for Custom Role Changes |
| 3.10.5 | Google | Ensure That Audit Logs are retained for a Minimum of 90 Days |
| 3.10.6 | AWS | Ensure That Audit Logs are retained for a Minimum of 90 Days |
| 3.10.7 | Azure | Ensure That Audit Logs are retained for a Minimum of 90 Days |

## 3.11 Collect Detailed Audit Logs

### Description

Configure detailed audit logging for enterprise assets containing confidential data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.

### Rationale

Detailed logs with timestamps provide a record of user activity, system events, and application actions. This allows administrators to identify suspicious activity, potential security breaches, and unauthorized access attempts.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 3.11.1 | AWS | Ensure CloudTrail is enabled in all regions |
| 3.11.2 | AWS | Ensure CloudTrail trails are integrated with CloudWatch Logs |
| 3.11.3 | Azure | Ensure that Azure Monitor Resource Logging is Enabled for All Services that Manage, Store, or Secure Confidential Data |
| 3.11.4 | Azure | Ensure that logging for Azure Key Vault is 'Enabled' |
| 3.11.5 | Azure | Ensure that Activity Log Alert exists for Create Policy Assignment |
| 3.11.6 | Azure | Ensure that Activity Log Alert exists for Delete Policy Assignment |
| 3.11.7 | Azure | Ensure that Activity Log Alert exists for Create or Update Network Security Group |
| 3.11.8 | Azure | Ensure that Activity Log Alert exists for Delete Network Security Group |
| 3.11.9 | Azure | Ensure that Activity Log Alert exists for Create or Update Security Solution |

| Spec | Platform | Description |
|---|---|---|
| 3.11.10 | Azure | Ensure that Activity Log Alert exists for Delete Security Solution |
| 3.11.11 | Azure | Ensure that Activity Log Alert exists for Create or Update SQL Server Firewall Rule |
| 3.11.12 | Azure | Ensure that Activity Log Alert exists for Delete SQL Server Firewall Rule |
| 3.11.13 | Azure | Ensure that Activity Log Alert exists for Create or Update Public IP Address rule |
| 3.11.14 | Azure | Ensure that Activity Log Alert exists for Delete Public IP Address rule |

# 4 Networking

## 4.1 Encrypt Confidential Data in Transit

### Description

Encrypt confidential data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).

### Rationale

Encryption protects confidential data when transmitted over untrusted network connections.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 4.1.1 | Google | Ensure No HTTPS or SSL Proxy Load Balancers Permit SSL Policies With Weak Cipher Suites |

## 4.2 Establish and Maintain a Secure Configuration Process for Network Infrastructure

### Description

Establish and maintain a secure configuration process for network devices. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

This CIS Control provides guidance for securing hardware and software. As delivered by the CSP, the default configurations for operating systems and applications are normally geared toward ease-of-deployment and ease-of-use -- not security. Basic controls, open services and ports, default accounts or passwords, older (vulnerable) protocols, pre-installation of unneeded software -- all can be exploitable in their default state. Even if a strong initial configuration is developed and deployed in the cloud, it must be continually managed to avoid configuration drift as software is updated or patched, new security vulnerabilities are reported, and configurations are “tweaked” to allow the installation of new software or to support new operational

requirements. If not, attackers will find opportunities to exploit both network- accessible services and client software.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 4.2.1 | Google | Ensure Legacy Networks Do Not Exist for Older Projects |
| 4.2.2 | Google | Ensure That DNSSEC Is Enabled for Cloud DNS |
| 4.2.3 | Google | Ensure That RSASHA1 Is Not Used for the Key-Signing Key in Cloud DNS DNSSEC |
| 4.2.4 | Google | Ensure That RSASHA1 Is Not Used for the Zone-Signing Key in Cloud DNS DNSSEC |
| 4.2.5 | AWS | Ensure that EC2 Metadata Service only allows IMDSv2 |

## 4.3 Implement and Manage a Firewall on Servers

### Description

Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.

### Rationale

Firewalls help to prevent unauthorized users from accessing servers or sending malicious payloads to those servers.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 4.3.1 | Azure | Ensure that RDP access from the Internet is evaluated and restricted |
| 4.3.2 | Azure | Ensure that SSH access from the Internet is evaluated and restricted |
| 4.3.3 | Google | Ensure That SSH Access Is Restricted From the Internet |
| 4.3.4 | Google | Ensure That RDP Access Is Restricted From the Internet |
| 4.3.5 | AWS | Ensure no Network ACLs allow ingress from 0.0.0.0/0 to remote server administration ports |
| 4.3.6 | AWS | Ensure no security groups allow ingress from 0.0.0.0/0 to remote server administration ports |
| 4.3.7 | AWS | Ensure no security groups allow ingress from ::/0 to remote server administration ports |

# 5 Storage

## 5.1 Establish and Maintain a Data Recovery Process

### Description

Establish and maintain a data recovery process. In the process, address the scope of data recovery activities, recovery prioritization, and the security of backup data. Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

Organizations need to establish and maintain data recovery practices sufficient to restore in-scope enterprise assets to a pre-incident and trusted state.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.1.1 | Azure | Ensure Soft Delete is Enabled for Azure Containers and Blob Storage |

## 5.2 Establish and Maintain a Secure Network Architecture

### Description

Establish and maintain a secure network architecture. A secure network architecture must address segmentation, least privilege, and availability, at a minimum.

### Rationale

Malicious actors can exploit insecure services, poor firewall and network configurations, and default or legacy credentials.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.2.1 | Azure | Ensure Default Network Access Rule for Storage Accounts is Set to Deny |

## 5.3 Encrypt Confidential Data in Transit

### Description

Encrypt confidential data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).

### Rationale

Encryption protects confidential data when transmitted over untrusted network connections.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.3.1 | Azure | Ensure that 'Secure transfer required' is set to 'Enabled' |
| 5.3.2 Azure | Ensure the "Minimum TLS version" for storage accounts is set to "Version 1.2" | |

## 5.4 Encrypt Confidential Data at Rest

### Description

Encrypt confidential data at rest on servers, applications, and databases. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.

### Rationale

Encryption at rest protects against some risks of unauthorized access to data, for example insecure disposal and reuse of storage media.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.4.1 | AWS | Ensure EBS Volume Encryption is Enabled in all Regions |
| 5.4.2 | AWS | Ensure that encryption is enabled for EFS file systems |

## 5.5 Configure Data Access Control Lists

### Description

Configure data access control lists based on a user’s need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.

### Rationale

The principle of least privilege reduces the risk of unauthorized actions being taken in your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.5.1 | AWS | Ensure that S3 Buckets are configured with 'Block public access (bucket settings)' |
| 5.5.2 | Azure | Ensure that 'Public access level' is disabled for storage accounts with blob containers |
| 5.5.3 | Google | Ensure That Cloud Storage Bucket Is Not Anonymously or Publicly Accessible |

## 5.6 Establish and Maintain a Secure Configuration Process

### Description

Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

"This CIS Control provides guidance for securing hardware and software. As delivered by the CSP, the default configurations for operating systems and applications are normally geared toward ease-of-deployment and ease-of-use -- not security. Basic controls, open services and ports, default accounts or passwords, older (vulnerable) protocols, pre-installation of unneeded software -- all can be exploitable in their default state. Even if a strong initial configuration is developed and deployed in the cloud, it must be continually managed to avoid configuration drift as software is updated or patched, new security vulnerabilities are reported, and configurations are “tweaked” to allow the installation of new software or to support new operational requirements. If not, attackers will find opportunities to exploit both network- accessible services and client software."

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.6.1 | Azure | Ensure that 'Enable key rotation reminders' is enabled for each Storage Account |

## 5.7 Securely Manage Enterprise Assets and Software

### Description

Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.

### Rationale

Secure management of assets and software guards against malicious users from being able to observe administrative communications with remote servers, possibly leading to compromise of that server, or from making configuration changes to introduce a security vulnerability into the server.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.7.1 | Azure | Ensure that Storage Account Access Keys are Periodically Regenerated |

## 5.8 Establish an Access Revoking Process

### Description

Establish and follow a process, preferably automated, for revoking access to enterprise assets, through disabling accounts immediately upon termination, rights revocation, or role change of a user. Disabling accounts, instead of deleting accounts, may be necessary to preserve audit trails.

### Rationale

Ensuring that access grants are revoked when they're no longer needed reduces the target area for malicious users.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 5.8.1 | Azure | Ensure that Shared Access Signature Tokens Expire Within an Hour |

# 6 Database Services

## 6.1 Use Standard Hardening Configuration Templates for Application Infrastructure

### Description

Use standard, industry-recommended hardening configuration templates for application infrastructure components. This includes underlying servers, databases, and web servers, and applies to cloud containers, Platform as a Service (PaaS) components, and SaaS components. Do not allow in-house developed software to weaken configuration hardening.

### Rationale

Industry-recommended hardening configuration templates reduce the attack surface area of your system and reduce the risk of configuration errors that could lead to a security incident.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.1.1 | Google | Ensure That the ‘Local_infile’ Database Flag for a Cloud SQL MySQL Instance Is Set to ‘Off’ |

## 6.2 Allowlist Authorized Scripts

### Description

Use technical controls, such as digital signatures and version control, to ensure that only authorized scripts, such as specific .ps1, .py, etc., files, are allowed to execute. Block unauthorized scripts from executing. Reassess bi-annually, or more frequently.

### Rationale

Unauthorized scripts can be used by malicious users to take over a system or take other destructive actions.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.2.1 | Google | Ensure 'external scripts enabled' database flag for Cloud SQL SQL Server instance is set to 'off' |

## 6.3 Encrypt Confidential Data in Transit

### Description

Encrypt confidential data in transit. Example implementations can include: Transport Layer Security (TLS) and Open Secure Shell (OpenSSH).

### Rationale

Encryption protects confidential data when transmitted over untrusted network connections.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.3.1 | Azure | Ensure 'Enforce SSL connection' is set to 'ENABLED' for PostgreSQL Database Server |
| 6.3.2 | Azure | Ensure 'Enforce SSL connection' is set to 'Enabled' for Standard MySQL Database Server |
| 6.3.3 | Azure | Ensure 'TLS Version' is set to 'TLSV1.2' for MySQL flexible Database Server |
| 6.3.4 | Google | Ensure That the Cloud SQL Database Instance Requires All Incoming Connections To Use SSL |

## 6.4 Encrypt Confidential Data at Rest

### Description

Encrypt confidential data at rest on servers, applications, and databases. Storage-layer encryption, also known as server-side encryption, meets the minimum requirement of this Safeguard. Additional encryption methods may include application-layer encryption, also known as client-side encryption, where access to the data storage device(s) does not permit access to the plain-text data.

### Rationale

Encryption at rest protects against some risks of unauthorized access to data, for example insecure disposal and reuse of storage media.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.4.1 | AWS | Ensure that encryption-at-rest is enabled for RDS Instances |
| 6.4.2 | Azure | Ensure that 'Data encryption' is set to 'On' on a SQL Database |

## 6.5 Configure Data Access Control Lists

### Description

Configure data access control lists based on a user's need to know. Apply data access control lists, also known as access permissions, to local and remote file systems, databases, and applications.

### Rationale

The principle of least privilege reduces the risk of unauthorized actions being taken in your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.5.1 | AWS | Ensure that public access is not given to RDS Instance |
| 6.5.2 | Azure | Ensure no Azure SQL Databases allow ingress from 0.0.0.0/0 (ANY IP) |
| 6.5.3 | Google | Ensure That Cloud SQL Database Instances Do Not Implicitly Whitelist All Public IP Addresses |
| 6.5.4 | Google | Ensure 'Skip_show_database' Database Flag for Cloud SQL MySQL Instance Is Set to 'On' |
| 6.5.5 | Google | Ensure that the 'cross db ownership chaining' database flag for Cloud SQL SQL Server instance is set to 'off' |
| 6.5.6 | Google | Ensure that the 'contained database authentication' database flag for Cloud SQL on the SQL Server instance is set to 'off' |

## 6.6 Establish and Maintain a Secure Configuration Process

### Description

Establish and maintain a secure configuration process for enterprise assets (end-user devices, including portable and mobile, non-computing/IoT devices, and servers) and software (operating systems and applications). Review and update documentation annually, or when significant enterprise changes occur that could impact this Safeguard.

### Rationale

This CIS Control provides guidance for securing hardware and software. As delivered by the CSP, the default configurations for operating systems and applications are normally geared toward ease-of-deployment and ease-of-use -- not security. Basic controls, open services and ports, default accounts or passwords, older

(vulnerable) protocols, pre-installation of unneeded software -- all can be exploitable in their default state. Even if a strong initial configuration is developed and deployed in the cloud, it must be continually managed to avoid configuration drift as software is updated or patched, new security vulnerabilities are reported, and configurations are “tweaked” to allow the installation of new software or to support new operational requirements. If not, attackers will find opportunities to exploit both network- accessible services and client software.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.6.1 | Google | Ensure 'user options' database flag for Cloud SQL SQL Server instance is not configured |
| 6.6.2 | Google | Ensure '3625 (trace flag)' database flag for all Cloud SQL Server instances is set to 'on' |

## 6.7 Implement and Manage a Firewall on Servers

### Description

Implement and manage a firewall on servers, where supported. Example implementations include a virtual firewall, operating system firewall, or a third-party firewall agent.

### Rationale

Firewalls help to prevent unauthorized users from accessing servers or sending malicious payloads to those servers.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.7.1 | Azure | Ensure 'Allow access to Azure services' for PostgreSQL Database Server is disabled |

## 6.8 Securely Manage Enterprise Assets and Software

### Description

Securely manage enterprise assets and software. Example implementations include managing configuration through version-controlled-infrastructure-as-code and accessing administrative interfaces over secure network protocols, such as Secure Shell (SSH) and Hypertext Transfer Protocol Secure (HTTPS). Do not use insecure management protocols, such as Telnet (Teletype Network) and HTTP, unless operationally essential.

### Rationale

Secure management of assets and software guards against malicious users from being able to observe administrative communications with remote servers, possibly leading to compromise of that server, or from making configuration changes to introduce a security vulnerability into the server.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.8.1 | Google | Ensure Instance IP assignment is set to private |

## 6.9 Manage Default Accounts on Enterprise Assets and Software

### Description

Manage default accounts on enterprise assets and software, such as root, administrator, and other pre-configured vendor accounts. Example implementations can include: disabling default accounts or making them unusable.

### Rationale

Products typically ship with insecure defaults that, if not configured securely, can be used by malicious users to take over a system.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.9.1 | Google | Ensure That a MySQL Database Instance Does Not Allow Anyone To Connect With Administrative Privileges |

## 6.10 Uninstall or Disable Unnecessary Services on Enterprise Assets and Software

### Description

Uninstall or disable unnecessary services on enterprise assets and software, such as an unused file sharing service, web application module, or service function.

### Rationale

Uninstalling and disabling unnecessary services reduces the target area of your systems.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.10.1 | Google | Ensure 'remote access' database flag for Cloud SQL SQL Server instance is set to 'off' |

### 6.11 Centralize Account Management

### Description

Centralize account management through a directory or identity service.

### Rationale

Centralizing makes administration simpler and therefore reduces risks related to unauthorized account creation or usage.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.11.1 | Azure | Ensure that Azure Active Directory Admin is Configured for SQL Servers |

## 6.12 Perform Automated Application Patch Management

### Description

Perform application updates on enterprise assets through automated patch management on a monthly, or more frequent, basis.

### Rationale

Patching remediates known vulnerabilities. Using automation makes this process routine and reduces the window of opportunity for attackers.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.12.1 | AWS | Ensure Auto Minor Version Upgrade feature is Enabled for RDS Instances |

## 6.13 Collect Audit Logs

### Description

Collect audit logs. Ensure that logging, per the enterprise’s audit log management process, has been enabled across enterprise assets and that logs are retained for at least a minimum period of time.

### Rationale

Having log files of what actions have taken place by users and also system events is fundamental to being able to detect security events.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.13.1 | Azure | Ensure Server Parameter 'log_checkpoints' is set to 'ON' for PostgreSQL Database Server |
| 6.13.2 | Azure | Ensure server parameter 'log_connections' is set to 'ON' for PostgreSQL Database Server |

| Spec | Platform | Description |
|---|---|---|
| 6.13.3 | Azure | Ensure server parameter 'log_disconnections' is set to 'ON' for PostgreSQL Database Server |

## 6.14 Ensure Adequate Audit Log Storage

### Description

Ensure that logging destinations maintain adequate storage to comply with the enterprise’s audit log management process.

### Rationale

Once configured, logs may generate large volumes of data. Organizations must ensure that logs are preserved according to the organization's retention policy and that there is sufficient storage for this requirement.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.14.1 | Azure | Ensure Server Parameter 'log_retention_days' is greater than 3 days for PostgreSQL Database Server |

## 6.15 Collect Detailed Audit Logs

### Description

Configure detailed audit logging for enterprise assets containing confidential data. Include event source, date, username, timestamp, source addresses, destination addresses, and other useful elements that could assist in a forensic investigation.

### Rationale

Detailed logs with timestamps provide a record of user activity, system events, and application actions. This allows administrators to identify suspicious activity, potential security breaches, and unauthorized access attempts.

### Audit

| Spec | Platform | Description |
|---|---|---|
| 6.15.1 | Azure | Ensure that 'Auditing' is set to 'On' |
| 6.15.2 | Google | Ensure That the ‘Log_connections’ Database Flag for Cloud SQL PostgreSQL Instance Is Set to ‘On’ |
| 6.15.3 | Google | Ensure That the ‘Log_disconnections’ Database Flag for Cloud SQL PostgreSQL Instance Is Set to ‘On’ |

| Spec | Platform | Description |
|---|---|---|
| 6.15.4 | Google | Ensure that the ‘Log_min_messages’ Flag for a Cloud SQL PostgreSQL Instance is set at minimum to 'Warning' |
| 6.15.5 | Google | Ensure ‘Log_min_error_statement’ Database Flag for Cloud SQL PostgreSQL Instance Is Set to ‘Error’ or Stricter |
| 6.15.6 | Google | Ensure That the ‘Log_min_duration_statement’ Database Flag for Cloud SQL PostgreSQL Instance Is Set to ‘-1’ (Disabled) |
| 6.15.7 | Google | Ensure That 'cloudsql.enable_pgaudit' Database Flag for each Cloud Sql Postgresql Instance Is Set to 'on' For Centralized Logging |
| 6.15.8 | AWS | Database logging should be enabled |