CIS Center for Internet Security®

CIS Benchmarks™

CIS API Security Guide

v1.0 - 08-05-2024

Terms of Use

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

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

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

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

Table of Contents

Overview

All CIS Benchmarks™ focus on technical configuration settings used to maintain and/or increase the security of the addressed technology, and they should be used in **conjunction** with other essential cyber hygiene tasks like:

- Monitoring the base operating system and applications for vulnerabilities and quickly updating with the latest security patches.
- End-point protection (Antivirus software, Endpoint Detection and Response (EDR), etc.).
- Logging and monitoring user and system activity.

In the end, the CIS Benchmarks™ are designed to be a key **component** of a comprehensive cybersecurity program.

Important Usage Information

All CIS Benchmarks™ are available free for non-commercial use from the CIS Website. They can be used to ***manually*** assess and remediate systems and applications. In lieu of manual assessment and remediation, there are several tools available to assist with assessment:

- CIS Configuration Assessment Tool (CIS-CAT® Pro Assessor)
- CIS Benchmarks™ Certified 3rd Party Tooling

These tools make the hardening process much more scalable for large numbers of systems and applications.

NOTE: Some tooling focuses only on the CIS Benchmarks™ Recommendations that can be fully automated (skipping ones marked **Manual**). It is important that ***ALL*** Recommendations (**Automated** and **Manual**) be addressed, since all are important for properly securing systems and are typically in scope for audits.

In addition, CIS has developed CIS Build Kits for some common technologies to assist in applying CIS Benchmarks™ Recommendations.

When remediating systems (changing configuration settings on deployed systems as per the CIS Benchmarks™ Recommendations), please approach this with caution and test thoroughly.

The following is a reasonable remediation approach to follow:

1. **NEVER** deploy a CIS Build Kit, or any internally developed remediation method, to production systems without proper testing.
2. Proper testing consists of the following:

a. Understand the configuration (including installed applications) of the targeted systems.
b. Read the Impact section of the given Recommendation to help determine if there might be an issue with the targeted systems.
c. Test the configuration changes on representative lab system(s). This way if there is some issue it can be resolved prior to deploying to any production systems.
d. When confident, initially deploy to a small sub-set of users and monitor closely for issues. This way if there is some issue it can be resolved prior to deploying more broadly.
e. When confident, iteratively deploy to additional groups and monitor closely for issues until deployment is complete. This way if there is some issue it can be resolved prior to continuing deployment.

NOTE: CIS and the CIS Benchmarks™ development communities in CIS WorkBench do their best to test and have high confidence in the Recommendations, but they cannot test potential conflicts with all possible system deployments. Known potential issues identified during CIS Benchmarks™ development are documented in the Impact section of each Recommendation.

By using CIS and/or CIS Benchmarks™ Certified tools, and being careful with remediation deployment, it is possible to harden large numbers of deployed systems in a cost effective, efficient, and safe manner.

NOTE: As previously stated, the PDF versions of the CIS Benchmarks™ are available for free, non-commercial use on the CIS Website. All other formats of the CIS Benchmarks™ (MS Word, Excel, and Build Kits) are available for CIS SecureSuite® members.

CIS-CAT® Pro is also available to CIS SecureSuite® members.

Target Technology Details

This document provides prescriptive guidance for establishing a secure configuration posture for Application Programming Interface (API) security. To obtain the latest version of this guide, please visit www.cisecurity.org. If you have questions, comments, or have identified ways to improve this guide, please write to us at support@cisecurity.org.

The Goal of the Guide

The goal of this guide is to provide comprehensive recommendations and illustrate best practices to ensure the security and integrity of APIs.
APIs serve as critical connectors in modern applications, facilitating communication and data exchange between different systems and services. Due to their importance APIs are increasingly becoming targets for cyberattacks which, when successful, commonly lead to significant data leaks with large footprints and collateral damage across the application ecosystem.
Securing APIs is foundational to protecting the sensitive data moving between applications and to ensuring the smooth operation of the application ecosystem as a whole.

How Should This Guide be Used?

This guide is designed to be a practical resource for creating a security strategy. It should be used when designing and implementing security controls and practices at various stages of the API lifecycle and across the different technologies used to build and operate APIs. APIs are by nature software that is running on a network and often runs on cloud resources, and this guide focuses on aspects in each of those areas that are API specific.

Intended Audience

This document is intended for API developers, DevOps engineers, application security administrators, security specialists, and anyone involved in the development, deployment, and maintenance of APIs.

Consensus Guidance

This CIS Benchmark™ was created using a consensus review process comprised of a global community of subject matter experts. The process combines real world experience with data-based information to create technology specific guidance to assist users to secure their environments. Consensus participants provide perspective from a diverse set of backgrounds including consulting, software development, audit and compliance, security research, operations, government, and legal.

Each CIS Benchmark undergoes two phases of consensus review. The first phase occurs during initial Benchmark development. During this phase, subject matter experts convene to discuss, create, and test working drafts of the Benchmark. This discussion occurs until consensus has been reached on Benchmark recommendations. The second phase begins after the Benchmark has been published. During this phase, all feedback provided by the Internet community is reviewed by the consensus team for incorporation in the Benchmark. If you are interested in participating in the consensus process, please visit https://workbench.cisecurity.org/.

Typographical Conventions

The following typographical conventions are used throughout this guide:

Convention	Meaning
`Stylized Monospace font`	Used for blocks of code, command, and script examples. Text should be interpreted exactly as presented.
`Monospace font`	Used for inline code, commands, UI/Menu selections or examples. Text should be interpreted exactly as presented.
`<Monospace font in brackets>`	Text set in angle brackets denote a variable requiring substitution for a real value.
Italic font	Used to reference other relevant settings, CIS Benchmarks and/or Benchmark Communities. Also, used to denote the title of a book, article, or other publication.
Bold font	Additional information or caveats things like **Notes**, **Warnings**, or **Cautions** (usually just the word itself and the rest of the text normal).

Recommendation Definitions

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

Title

Concise description for the recommendation's intended configuration.

Assessment Status

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

Automated

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

Manual

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

Profile

A collection of recommendations for securing a technology or a supporting platform. Most benchmarks include at least a Level 1 and Level 2 Profile. Level 2 extends Level 1 recommendations and is not a standalone profile. The Profile Definitions section in the benchmark provides the definitions as they pertain to the recommendations included for the technology.

Description

Detailed information pertaining to the setting with which the recommendation is concerned. In some cases, the description will include the recommended value.

Rationale Statement

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

Impact Statement

Any security, functionality, or operational consequences that can result from following the recommendation.

Audit Procedure

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

Remediation Procedure

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

Default Value

Default value for the given setting in this recommendation, if known. If not known, either not configured or not defined will be applied.

References

Additional documentation relative to the recommendation.

CIS Critical Security Controls® (CIS Controls®)

The mapping between a recommendation and the CIS Controls is organized by CIS Controls version, Safeguard, and Implementation Group (IG). The Benchmark in its entirety addresses the CIS Controls safeguards of (v7) “5.1 - Establish Secure Configurations” and (v8) '4.1 - Establish and Maintain a Secure Configuration Process” so individual recommendations will not be mapped to these safeguards.

Additional Information

Supplementary information that does not correspond to any other field but may be useful to the user.

Profile Definitions

The following configuration profiles are defined by this Benchmark:

- **Level 1**
- **Level 2 - Master Node**

Acknowledgements

This Benchmark exemplifies the great things a community of users, vendors, and subject matter experts can accomplish through consensus collaboration. The CIS community thanks the entire consensus team with special recognition to the following individuals who contributed greatly to the creation of this guide:

Contributor
Viktor Markopoulos
Deborah Broomfield
Phil White , Center for Internet Security, New York
Timo Ruppell

Recommendations

1 Design

1.1 Documentation

This section consists of API documentation recommendations and the role it plays in API security. Establishing a standardized framework for documenting all aspects of the API ensures clarity and consistency across teams (developers, consumers, stakeholders).

1.1.1 Define a Format (Manual)

Profile Applicability:

- Level 1

Description:

Establish a comprehensive standard framework for documenting all aspects of the APIs

Rationale:

Having a standardized documentation framework for APIs offers clarity and consistency, serving as a single source of truth for all stakeholders involved. Having this resource ensures accuracy and also promotes collaboration within and between teams.

Audit:

- Review the documentation requirements.
- Assess the comprehensiveness of the documentation.
- Monitor all API changes and edit the documentation accordingly.

Remediation:

- Complete a gap analysis between the existing documentation and the desired standards or requirements. This review will pinpoint areas that are incomplete, outdated, inaccurate, or lacking in coverage and highlights areas for improvement.
- Review feedback from impacted teams, and implement the feedback if necessary.
- Add versioning, if required.
- Add all missing information into the documentation.
- Implement an ongoing monitoring procedure for the documentation, including putting in place mechanisms to enable continuous updates.

1.1.2 Specify Endpoints (Manual)

Profile Applicability:

- Level 1

Description:

The documentation should include detailed definitions and descriptions of the API endpoints.

Rationale:

Clear endpoint specifications in API documentation enhance development efficiency and security by outlining the API's capabilities, including endpoints, request formats, methods, and responses. This ensures developers interact with APIs accurately, reducing errors and vulnerabilities. A concise endpoint summary aids end-users in understanding the API's functionality and finding relevant endpoints quickly. Use cases and examples enhance comprehension and facilitate API integration into workflows.

Audit:

- Review the documentation requirements.
- Review the specification files, including all existing endpoints utilized by the API.
- Get feedback from consumers, stakeholders, and developers.

Remediation:

- Perform a gap analysis on the current state of the specification files and the desired standards or requirements. This review will pinpoint areas that are incomplete, outdated, inaccurate, or lacking in coverage and highlights areas for improvement.
- Update documentation requirements, if needed.
- Add missing endpoints from already-existing or updated requirements.
- Implement a continuous monitoring and evaluation procedure.

1.1.3 Specify Methods (Manual)

Profile Applicability:

- Level 1

Description:

The documentation should define the resources the API offers and detail the methods used for each endpoint.

Rationale:

Specifying methods in API documentation offers clarity for the development team by detailing supported HTTP methods, helping developers understand intended actions for each endpoint, such as retrieving data or modifying resources. This reduces trial and error, ensuring correct API implementation and minimizing security risks associated with inappropriate method usage. For end-users, the documentation provides a concise overview of supported methods, enabling them to quickly grasp available actions.

Audit:

- Review the documentation requirements.
- Assess the specification files to determine the methods used in the existing endpoints that are utilized by the API.
- Check that methods follow a specific pattern based on their design type (e.g. CRUD for REST).
- Gather feedback from consumers, stakeholders, and developers.

Remediation:

- Perform a gap analysis on the current state of the specification files and the desired standards or requirements to identify any discrepancies. This review will pinpoint areas that are incomplete, outdated, inaccurate, or lacking in coverage and highlights areas for improvement.
- Gather feedback from all interested parties including consumers, stakeholders, and developers, to understand their needs and experiences.
- Update documentation requirements, if needed.
- Add or update missing methods from already-existing or updated requirements.
- Implement a continuous monitoring and evaluation procedure.

1.1.4 Specify Data Models (Manual)

Profile Applicability:

- Level 1

Description:

The documentation should specify the data models and objects that are being used in the API.

Rationale:

Detailing data models in API documentation provides developers with clarity on the data structure and format for requests and responses, leading to more efficient and secure development. Clear data models enable accurate request formatting and response handling, reducing errors and security vulnerabilities, such as Insecure Direct Object Reference (IDOR). Specifying data models also helps end-users understand the types of data they can provide or expect, enhancing usability and adoption.

Audit:

- Review the documentation requirements.
- Evaluate the specification files to ensure they include all models and variables utilized by the API.
- Gather feedback from consumers, stakeholders, and developers.

Remediation:

- Perform a gap analysis on the current state of the specification files and the desired standard to identify any discrepancies.
- Gather feedback from consumers, stakeholders, and developers to understand their needs and perspectives.
- Update documentation requirements, if needed.
- Add any missing data models or variables that are required based on existing or updated requirements.
- Implement a continuous monitoring and evaluation procedure.

1.1.5 Select an Automation-Friendly Format (Manual)

Profile Applicability:

- Level 1

Description:

API documentation should be in an automation-friendly format for easy use and integration with automation tools, supporting efficient management and automation of API interactions to enhance development processes and productivity.

Rationale:

By adopting an automation-friendly format, updates can be automatically pulled directly from development sources like Git commits, ensuring that documentation stays current with minimal manual effort. This automation saves time and streamlines the documentation process, allowing developers to focus more on coding and less on manual documentation tasks.

Audit:

Remediation:

- Decide on a format that can be used by automation programs e.g. RAML (RESTful API Modeling Language), OpenAPI, and API Blueprint.
- Make any manual changes that are required first.
- Implement automation scripts within the development pipeline to automatically update the documentation.
- Implement syntactic validity checks to ensure that updates to the documentation adhere to specified standards and formatting requirements.

1.1.6 Specify Endpoint Parameters (Manual)

Profile Applicability:

- Level 1

Description:

Specifying endpoints in API documentation involves detailing the parameters utilized by each endpoint, including the type of data expected in requests, whether parameters are required or optional, and the content type used for parameter values.

Rationale:

Specifying endpoints in API documentation provides clarity to the development team, leading to more efficient and secure development practices. It helps developers understand how to interact with the API correctly, identify potential security risks related to parameters, and implement proper validation to mitigate risks like injection attacks. This documentation ensures consistent usage of endpoints and reduces the likelihood of errors or vulnerabilities during API development and implementation.

Audit:

- Review each endpoint to identify specified parameters used in API requests.
- Determine if parameters are required or optional for each endpoint, including special cases like sorting parameters.
- Review the documentation requirements.
- Assess the specification files, including all required parameters, their data type, and their content type.
- Gather feedback from interested parties.

Remediation:

- Perform a gap analysis on the current state of the specification files and the desired standard.
- Update documentation requirements, if needed.
- Document the data types expected for each parameter passed in requests.
- Specify the content type (e.g., application/json) for parameters used in API requests.
- Add missing parameters from already-existing or updated requirements.
- Implement a continuous monitoring and evaluation procedure.

1.1.7 Make Documentation the Single Source of Truth (Manual)

Profile Applicability:

- Level 1

Description:

The API documentation should serve as a single point of truth, providing one comprehensive reference document for all teams to consult.

Rationale:

Having multiple sources of documentation can result in inconsistencies and miscommunication among teams, leading to poor coordination. A unified document ensures that all teams have access to the same information, promoting alignment and clarity across the organization.

Audit:

Ensure only one document exists on an endpoint or location that all teams can access.

Remediation:

Create an endpoint that all teams can access and upload the documentation there.

1.1.8 Establish Clear and Comprehensive Documentation Standards (Manual)

Profile Applicability:

- Level 1

Description:

The documentation should align with industry-standard frameworks that all teams can easily understand and follow. The documentation must encompass all essential information needed by each team.

Rationale:

If the documentation is comprehensible by all teams, the chance of miscommunication is minimized.

Audit:

Gather feedback from all teams to find out if the current framework being used is understandable.

Remediation:

- Find a format that all teams understand and are familiar with.
- Address any identified gaps or areas of confusion in the documentation based on the feedback received.
- Update the documentation accordingly to ensure it remains comprehensive and clear.
- Communicate any revisions or updates to the teams.
- Encourage ongoing feedback and collaboration to continuously improve the documentation and support the team's productivity.

1.1.9 Communicate Documentation Updates Across all Teams (Manual)

Profile Applicability:

- Level 1

Description:

Make sure that any changes to the documentation is communicated to all teams involved.

Rationale:

Teams should be made aware of all changes made so that they work on the same version of the documentation.

Audit:

- Check that the communication protocols for documentation changes are working properly to notify all teams.
- Gather feedback from the teams to confirm that the documentation provides sufficient information for them to work effectively.

Remediation:

Create an automation to notify all involved teams when changes are made to the documentation.

1.2 Authentication and Authorization

This section consists of recommendations on authentication and authorization mechanisms as well as defining the various technologies used.

1.2.1 Define Authentication and Authorization Requirements (Manual)

Profile Applicability:

- Level 1

Description:

Decide if the API requires authentication and authorization mechanisms along with deciding on the tools to be used. Role-based access controls will be implemented with user roles defined as user, admin, and superadmin, each having specific permissions and access levels to different data and functions within the API.

Rationale:

Defining API authentication and authorization requirements provides a structured plan from the outset, outlining how user identity and access will be managed, which helps with risk mitigation by implementing appropriate security controls. This approach ensures compliance with regulatory laws and standards governing data protection and privacy, safeguarding against potential legal and compliance risks associated with API usage.

Audit:

- Review the documentation and the requirements provided by the development team and the compliance standards to ensure alignment and completeness.
- Evaluate compliance with requirements using VAPT and/or automated solutions across various use cases, including normal scenarios and situations where low-privilege users attempt to access high-privilege data or data belonging to other users.
- Ensure the current state complies with compliance standards.
- Ensure proper and secure management of secrets and tokens, employing solutions like encrypted vaults to safeguard sensitive information.

Remediation:

- Assess current authentication and authorization mechanisms.
- Conduct a gap analysis between the existing mechanisms and the compliance standards' mechanisms.
- Define authentication methods (for example, SSO, OAuth, API key, JWT).
- Establish user and entity-specific authorization controls (for example, user, admin).
- Document requirements.

1.2.2 Determine API Authorization Approach (Manual)

Profile Applicability:

- Level 1

Description:

Decide if the API will be private or public, internal or external.

Rationale:

The decision will determine what kind of authorization and visibility the API will have.

Audit:

Check that the documentation has the relevant authorization information on each API.

Remediation:

- Evaluate API user access requirements.
- Determine API authorization requirements for storing user-sensitive data.
- Document all changes and decisions made.

1.2.3 Define User Roles (Manual)

Profile Applicability:

- Level 1

Description:

Determine the specific user roles that should exist and define the hierarchy of the roles.

Rationale:

Having clearly defined user roles establishes a clear plan for how users will interact with the API and outlines the structure of the user base.

Audit:

Confirm that the documentation clearly states the roles that a user should have.

Remediation:

- Determine the intended users of the API.
- Select roles based on the nature of the API and its utility.
- Document the hierarchy and the roles.

1.2.4 Decide on role-based access controls (Manual)

Profile Applicability:

- Level 1

Description:

Determine the access and authorization levels associated with each role.

Rationale:

Each role should be assigned specific data access and functional permissions to prevent disruptions, privilege escalation, and unauthorized access.

Audit:

Check that the documentation states the access levels of each role.

Remediation:

- Determine the hierarchy of the roles.
- Define what data and functions each role will have.
- Document the decisions.

1.2.5 Choose Authentication Mechanisms (Manual)

Profile Applicability:

- Level 1

Description:

Having appropriate authentication mechanisms in place helps ensure secure access control and user authentication.

Rationale:

Decide on authentication mechanisms to ensure secure access control and protect sensitive data from unauthorized access or malicious activities. Implementing strong authentication mechanisms helps verify the identity of users or systems interacting with the API, mitigating risks associated with unauthorized access, data breaches, and potential security vulnerabilities.

Audit:

Consult the documentation on current authentication mechanisms.

Remediation:

- Select at least one authentication mechanism appropriate for the API usage.
- Document the decisions.

1.3 Data Security and Privacy

This section consists of recommendations to conduct data assessments for understanding the types of data utilized within the API, aiding in proactive planning for implementing appropriate data protection mechanisms.

1.3.1 Assess Data Protection Needs (Manual)

Profile Applicability:

- Level 1

Description:

Assessing data protection needs for APIs involves evaluating security requirements, identifying data types and vulnerabilities, and implementing measures like encryption and access controls to safeguard sensitive information.

Rationale:

Assessing data protection needs for APIs is crucial for several reasons. It involves identifying sensitive data and implementing security precautions like encryption to protect against unauthorized access. By understanding and addressing data protection needs, organizations can mitigate risks of data leaks and unauthorized access, enhancing trust and reputation with users and stakeholders. Additionally, this process enables informed, risk-based decision making by aligning security measures with identified risks and compliance requirements.

Audit:

- Conduct a thorough assessment of the data that is being handled (their sensitivity) by the API and of the data that is being stored.
- Make sure that encryption standards are being applied for data both in transit with SSL/TLS and at rest with encryption and hashing algorithms.
- Perform a risk analysis on the data for potential threats and vulnerabilities linked with the data in question.
- Validate the data protection measures.

Remediation:

- Review the data handled by the API.
- Conduct a thorough threat model.
- Implement encryption mechanisms for data both in rest and transit.

1.3.2 Determine Applicable Compliance Standards (Manual)

Profile Applicability:

- Level 1

Description:

Identify all legal and regulatory standards that apply to handling specific data. This requires understanding what rules and compliance standards an API must follow to be allowed to handle sensitive information.

Rationale:

Ensuring legal obligations are met by adhering to regulations like GDPR or HIPAA, reduces legal risks related to data handling. Meeting compliance standards prioritizes customer data protection, outlining measures such as encryption and access controls to safeguard sensitive information. These standards contribute to effective risk management by implementing protections that mitigate potential data breaches or unauthorized access, enhancing overall data security.

Impact:

Some standards require regular security-related activities such as penetration and compliance tests.

Audit:

- Identify the regulation(s) and compliance standard(s) your API falls under.
- Review the compliance standards.
- Assess how data is handled and stored according to the relevant regulations.

Remediation:

- Classify data and identify the type of data your API handles.
- Research regulatory laws that revolve around the specific data.
- Analyze the gap between the current state of the API and the compliance standard.
- Document steps.
- Continuous monitoring and review.

2 Develop

2.1 Identity and Access Management

2.1.1 Define User Identity Management Protocols within the System (Manual)

Profile Applicability:

- Level 1

Description:

Establish clear user identity management protocols.

Rationale:

Defining user identity protocols within a system is crucial for several reasons. It ensures a clear understanding of managing user identities, including authentication and authorization. These protocols aid in risk mitigation by enforcing strong access controls, reducing unauthorized access risks. Allocating resources to identity management maintains operational efficiency, scalability, and security compliance.

Audit:

- Review existing authentication policies.
- Review existing access control policies.
- Evaluate the registration and user information process.
- Evaluate the user information storage systems currently in place.
- Review logs.

Remediation:

- Enhance authentication mechanisms and management
- Document all changes made.
- Communicate with all teams, to make them aware of changes.
- Monitor the affected systems.

2.1.2 Define User Types and Required Information (Manual)

Profile Applicability:

- Level 1

Description:

Specify user types (for example, admin, read only user, superadmin) and their identification/registration details (username, email address, password, required MFA) for system setup.

Rationale:

Defining user types and required information establishes clear authorization levels and a structured hierarchy within a system. Categorizing users into roles like administrators, regular users, and service accounts allows for the proper assignment of permissions based on specific responsibilities. Specifying necessary user details such as email addresses and passwords ensures standardized and secure user registration processes.

Audit:

- Review the documentation for existing user types and their required information.
- Review all systems and code, establish who requires specific information and ensure that users cannot access accounts without meeting all necessary requirements.

Remediation:

- Implement any required changes.
- Communicate with all teams about the changes that have been made.
- Get relevant stakeholder feedback.
- Document all changes accordingly.

2.1.3 Define Authentication Methods and Protocols (Manual)

Profile Applicability:

- Level 1

Description:

Specify the authentication methods (credentials, tokens, use of third party) and protocols.

Rationale:

Authentication methods like password logins and multi-factor authentication (MFA) enhance security by requiring users to verify their identity with multiple factors.

Audit:

- Review the existing authentication policy and documentation.
- Assess if regulatory compliance is currently being met.
- Evaluate existing authentication methods being used.
- Review protocols, check that secure transmission is set.
- Evaluate encryption when data is at rest.
- Document findings and recommendations.

Remediation:

- Update the authentication policy and documentation.
- Ensure regulatory compliance measures are in place.
- Implement strong authentication methods.
- Implement secure transmission.
- Encrypt data at rest.
- Update the documentation accordingly.

2.1.4 Define Access Levels for Each Role (Manual)

Profile Applicability:

- Level 1

Description:

Different user types will have different levels of access and authorization requirements. These should be clearly defined.

Rationale:

Clearly defining levels of access and authorization for different user types is essential for maintaining security, ensuring regulatory compliance, and improving operational efficiency. It helps prevent unauthorized access, supports legal requirements, and enables users to perform their tasks effectively while minimizing risks.

Audit:

- Review existing access control policies.
- Assess regulatory compliance.
- Examine current user role permissions.

Remediation:

- Define roles based on role responsibilities.
- Configure user role permissions.
- Document all changes made.

2.1.5 Define Authorization Models to Enforce Access Restrictions (Manual)

Profile Applicability:

- Level 1

Description:

Define frameworks and systems that control and limit user access to resources based on their roles and permissions. Examples include Role-Based Access Control (RBAC), and Attribute-Based Access Control (ABAC).

Rationale:

Defining authorization models to enforce access restrictions and access control lists is essential for maintaining security and compliance within a system. By clearly specifying who can access sensitive information and perform specific actions, these models protect against unauthorized access and potential breaches.

Audit:

- Review existing authorization policy documentation.
- Assess if existing procedures are compliant with regulations.
- Evaluate the current implementation process.
- Evaluate the existing authorization models in place.
- Examine permissions of individual roles.
- Document findings and recommendations.

Remediation:

- Review authorization policy documentation.
- Evaluate regulatory compliance.
- Evaluate the implementation to be used (RBAC OR ABAC).
- Implement permissions of individual roles.
- Document all changes made.

2.1.6 Implement Cryptographic Techniques to Secure User Credentials and Authentication Tokens (Manual)

Profile Applicability:

- Level 1

Description:

Specify protocols to secure user data transmission and storage with cryptographic protocols.

Rationale:

Implementing cryptographic techniques to secure user credentials and authentication tokens helps mitigate risks and prevent unauthorized access. Encrypting passwords and tokens helps keep sensitive data secure and complies with regulations such as GDPR. This renders data unreadable to unauthorized parties, protecting user credentials and authentication tokens from potential breaches and ensuring compliance with data protection laws and industry standards.

Audit:

- Review existing relevant policies.
- Assess if existing procedures are compliant with regulations.
- Assess password hashing methods.
- Assess token storage and method of generation.
- Assess TLS/SSL versions.
- Assess key management practices and policies.
- Assess monitoring practices.

Remediation:

- Select password hashing methods.
- Choose encrypted token storage and a secure method of generation.
- Use TLS/SSL.
- Enhance key management practices and policies.
- Apply monitoring practices.
- Document all changes made.

2.2 API Endpoint Security

2.2.1 Identify the security requirements for each API endpoint (Manual)

Profile Applicability:

- Level 1

Description:

Specify security requirements for each API endpoint, including authentication methods (e.g., OAuth 2.0, API keys), encryption standards (e.g., TLS/SSL), authorization controls (e.g., role-based access), input validation, rate limiting, and logging, to mitigate risks.

Rationale:

Identifying the security requirements for each API endpoint allows for a risk assessment to be done, which will evaluate potential threats and vulnerabilities specific to each endpoint. This process ensures compliance with industry and regulatory standards, preventing legal issues. Additionally, it enables the implementation of tailored security controls that address the unique needs and risks of each endpoint.

Audit:

- Review all existing documentation.
- Assess compliance requirements to evaluate if there are any existing gaps.
- Perform risk assessment/Vulnerability Assessment and Penetration Testing (VAPT).
- Document all requirements and procedures.

Remediation:

- Update the documentation with all changes made.
- Implement security control based on the endpoint requirements.
- Review and update the security requirements.

2.2.2 Enforce TLS/SSL protocols (Manual)

Profile Applicability:

- Level 1

Description:

Enforce TLS and SSL protocols.

Rationale:

Enforcing TLS/SSL protocols ensures data transfer encryption, safeguarding against interception and eavesdropping by malicious actors. By preventing Man-in-the-Middle (MITM) attacks, TLS/SSL maintains data confidentiality and integrity during transmission. It also protects the authentication procedure from unauthorized access, meeting compliance requirements and ensuring secure communication channels between clients and servers.

Audit:

- Review the current configuration settings.
- Review certificates to ensure they are current, properly configured, and secure, preventing potential vulnerabilities.
- Review HTTPS implementation and versions.
- Review any non-compliant connections.
- Review policies and compliance requirements.

Remediation:

- Update the configuration settings.
- Renew all certificates.
- Implement HTTPS redirection from port 80 (HTTP) to port 443 (HTTPS) and implement HSTS to enforce HTTPS.
- Address any vulnerabilities introduced by old and current HTTPS versions.
- Enforce monitoring and alerting.
- Update the documentation.

2.2.3 Enforce input validation and sanitization (Manual)

Profile Applicability:

- Level 1

Description:

Enforce input validation and sanitization.

Rationale:

Enforce input validation and sanitization to shield against common attacks like SQL injection (SQLi) and cross-site scripting (XSS), which exploit vulnerable input fields. By dropping or sanitizing suspicious or malicious requests, it ensures data integrity and prevents unauthorized access to internal systems.

Audit:

- Review the codebase.
- Identify the input sources.
- Review validation rules.
- Check sanitization measures.
- Review error handling.
- Review logging and monitoring.

Remediation:

- Update input validation rules.
- Implement sanitization rules.
- Assign the input validation server-side.
- Handle errors.
- Perform a static analysis, by making use of a WAF.
- Document security controls.

2.2.4 Enforce Rate Limiting mechanisms (Manual)

Profile Applicability:

- Level 1

Description:

Enforce rate limiting mechanisms to provide protection from repeated requests that target an endpoint.

Rationale:

Enforcing rate limiting mechanisms helps prevent Distributed Denial of Service (DDoS) and brute-force attacks by limiting the number of requests an attacker can make within a given timeframe. This ensures the availability of services for legitimate users and protects server resources from being overwhelmed. Compliance requirements often mandate implementing such measures to enhance security. By maintaining Quality of Service (QoS), rate limiting ensures a consistent and reliable experience for users while mitigating the risk of abuse or exploitation of system resources.

Audit:

- Review the application architecture.
- Analyze existing rate-limiting configurations.
- Examine the logs.
- Review regulation compliance and check if it is currently in line with regulations.
- Evaluate Denial of Service (DoS) resilience.
- Perform stress tests on existing rate limiting mechanisms.

Remediation:

- Update rate-limiting configurations.
- Implement behavioral rate-limiting based on session and/or IP behavior.
- Enforce logging and monitoring procedures.
- Deploy distributed rate limiting (load balancers), CDNs, WAFs, IDS/IPS).
- Document all actions taken.

2.2.5 Define logging and monitoring procedures (Manual)

Profile Applicability:

- Level 1

Description:

Specify the locations and methods for logging, detailing what will be monitored and the procedures for monitoring them.

Rationale:

Defining logging and monitoring practices enhances system visibility, allowing for the identification of potential threats. Detailed logs support future analysis, whether for incident response or service improvement. Comprehensive logging also ensures compliance with regulations and aids in effective risk management by identifying and mitigating risks promptly.

Audit:

- Review the existing logging and monitoring policies.
- Assess the current methods used.
- Review logging confidentiality and access management to determine who is allowed to view the logs.
- Evaluate log retention.

Remediation:

- Update policies and procedures.
- Use the appropriate logging detail required for endpoints.
- Implement logging controls to protect log files against third-party access.
- Automate the logging procedure (collecting, analysis, retention, archiving).
- Regular review and testing.

2.2.6 Establish Practices Based on Regulations and Industry Standards (Manual)

Profile Applicability:

- Level 1

Description:

Put guidelines and procedures in place in accordance with regulatory requirements and industry standards to ensure compliance and best practices.

Rationale:

Defining practices dictated by regulations and industry standards ensures legal assurance, protects data and privacy, avoids financial losses from fines, and builds trust with customers. Compliance with these practices safeguards the organization against legal repercussions and enhances its reputation for reliability and integrity.

Audit:

- Review all standards impacting specific APIs, considering different standards based on the nature of the data the API handles.
- Perform a gap analysis to determine if existing procedures are in line with regulations.
- Review all existing documentation, policies, and procedures.
- Complete compliance assessments.
- Identify missing or faulty controls and mechanisms.
- Complete external (compliance) assessments.
- Risk management from the above-mentioned assessments.

Remediation:

- Identification of regulatory standards affecting the API.
- Implement missing (or faulty) controls and mechanisms.
- Assess changes internally and externally.
- Document all changes and updates.

2.3 Use Proper Error Handling

2.3.1 Identify possible error occurrences (Manual)

Profile Applicability:

- Level 1

Description:

Find all endpoints where errors may occur (e.g. based on user input).

Rationale:

Identifying possible error occurrences in APIs is essential to prevent unpredictable system behavior and avoid risks such as inadvertently disclosing internal paths and technologies used. By addressing these errors proactively, you enhance security and provide a more reliable user experience, ultimately contributing to a more robust and user-friendly application.

Audit:

- Review the API design documentation.
- Analyze user input endpoints and data sources.
- Analyze business logic.
- Review external dependencies and how they integrate with the API.
- Consider environmental factors like network latency, system load.

Remediation:

- Set a standardized way to present errors.
- Implement input validation and sanitization.
- Catch errors at code level.
- Enforce monitoring (error log).

2.3.2 Establish standardized error handling procedures (Manual)

Profile Applicability:

- Level 1

Description:

Put in place a consistent procedure to handle errors.

Rationale:

Establishing standardized error-handling procedures ensures consistency across the API, providing a uniform approach to managing and communicating errors. This improves the clarity and usefulness of error messages for developers and users, enhancing overall user experience. It also prevents revealing sensitive information that could help attackers.

Audit:

- Review the documentation by inspecting the existing error format that is in place.
- Identify potential disclosure of internal system information in error messages.
- Assess all existing error message quality - is it comprehensive, does it correctly communicate the error.
- Verify HTTP standards compliance, for example, 404 error should be used for non-existing resources, 401/403 error codes should be used for unauthorized access and so on.

Remediation:

- Improve the clarity of error messages.
- Establish guidelines for handling errors.
- Enforce HTTP return status compliance.
- Update the documentation accordingly.

2.3.3 Enforce error logging (Manual)

Profile Applicability:

- Level 1

Description:

Enforce error logging in API security to systematically record and store encountered errors, which will allow for future analysis.

Rationale:

Enforcing error logging in API security provides visibility into the system's response and state, aiding in the identification of errors or issues and helping to pinpoint potentially malicious activity. These logs also enable comprehensive future analysis, contributing to improved Quality of Service (QoS) by providing proactive measures to address and prevent recurring issues.

Audit:

- Review the existing logging policy.
- Review the implementation and the methods used for logging.
- Review storage methods and access level requirements.
- Review log retention policy.
- Test the logging functionality.

Remediation:

- Implement logging with a robust method, ideally avoiding third-party services.
- Configure retention policies.
- Adjust the granularity of logging.
- Store logs in a secure place.
- Implement access controls for stored logs.

2.3.4 Enforce error monitoring (Manual)

Profile Applicability:

- Level 1

Description:

Make logging a mandatory practice for all endpoints.

Rationale:

Enforcing error monitoring in APIs is essential for ensuring timely issue and threat detection, facilitating swift system recoverability, and complying with regulatory standards. Comprehensive error monitoring enables quick identification and remediation of issues, promoting system resilience and integrity. Additionally, adherence to regulation standards necessitates robust error monitoring practices to mitigate legal risk.

Audit:

- Review logging configurations and implementation.
- Review configuration compliance and coverage for all endpoints.

Remediation:

- Enable API and system logging.
- Implement a robust method for logging.
- Enforce configuration and logging settings to all endpoints to achieve 100% coverage.
- Regular configuration review and updates.

2.3.5 Establish a strategy for error recovery (Manual)

Profile Applicability:

- Level 1

Description:

Establish a standard procedure to recover from errors and system failures.

Rationale:

Establishing an error recovery strategy will minimize downtime, preserve data integrity, pinpoint root causes, and mitigating business risks. It enables swift issue resolution, preventing disruptions to operations, while also safeguarding data from loss or corruption. Furthermore, it aids in identifying the underlying causes of errors, facilitating more efficient troubleshooting.

Audit:

- Review existing procedure documentation.
- Assess error handling when error occurs.
- Check if a system recoverability procedure is in place.
- Ensure logs are being kept for post-recovery analysis.

Remediation:

- Document error recovery procedures.
- Implement automated recovery mechanisms.
- Establish operational mechanisms for teams involved.
- Establish monitoring and alerting mechanisms.
- Regularly test and validate procedures.

2.4 Session Management

2.4.1 Define the session lifecycle stages (Manual)

Profile Applicability:

- Level 1

Description:

Identify the various stages a user undergoes that occur during a session, this includes for example, creation, authentication, termination and so on.

Rationale:

Defining the session lifecycle stages is important because it provides a clear plan for dividing a session into distinct phases, allowing focused development of each stage. This clarity also aids in troubleshooting by making it easier to identify and address issues within specific parts of the session lifecycle.

Audit:

- Review documentation regarding the path of a user from user creation to logging out.
- Ensure that the codebase handles each stage separately.
- Ensure the logic behind the code follows a logical path, for example, active usage functions are permitted after authentication.

Remediation:

- Separate every stage based on specific actions. For example, registering is the creation stage, logout is the termination stage.
- Implement each function for every stage with separate logic.

2.4.2 Define how a session is initialized (Manual)

Profile Applicability:

- Level 1

Description:

Establish how an active user session is initiated.

Rationale:

Establishing how a user starts to have an active session ensures consistency in the activation process for all users, which is crucial for maintaining uniform security measures.

Audit:

- Review the existing documentation surrounding user initialization.
- Assess implementation consistency.
- Evaluate authentication mechanisms.
- Validate the session creation by the response (if a JWT is generated etc).

Remediation:

- Find the logical beginning of a session (for example, login).
- Develop a secure method for initializing a new session.
- Develop a validation mechanism for the initialized session.

2.4.3 Define session duration and validity (Manual)

Profile Applicability:

- Level 1

Description:

Establish a typical session duration to ensure the system can invalidate sessions in a timely manner.

Rationale:

Establishing a typical session duration is crucial for better system resource management, as it allows the system to efficiently handle and invalidate tokens, freeing up storage.It also helps prevent exposures, such as leaked tokens, and reduces the risk of unauthorized access by ensuring that hijacked sessions are invalidated promptly.

Audit:

- Evaluate session timeout duration.
- Assess renewal mechanisms.

Remediation:

- Decide on a session timeout setting based on security requirements (for example, risk exposure, session sensitivity).
- Implement a session timeout mechanism.
- Implement a session renewal mechanism.
- Communicate the policy to users.

2.4.4 Set up mechanisms to maintain sessions (Manual)

Profile Applicability:

- Level 1

Description:

Establish a method to manage and sustain active user sessions securely and efficiently based on user interaction. This involves implementing token-based authentication (like JWTs), session expiry and renewal rules, secure token storage, and procedures for invalidating and logging out sessions.

Rationale:

This enhances user experience by ensuring uninterrupted access to resources. It also prevents session hijacking by implementing robust authentication and authorization protocols, safeguarding user accounts and sensitive data.

Audit:

- Assess implementation consistency.
- Evaluate the session update frequency to ensure it is reasonable, as excessively long intervals can pose security risks.

Remediation:

- Standardize session maintenance practices.
- Implement secure session update mechanisms.
- Monitor active sessions.

2.4.5 Define session storage procedures (Manual)

Profile Applicability:

- Level 1

Description:

Establish a method for securely storing session-related data, such as user authentication tokens or session identifiers. Encrypt session-related data such as authentication tokens or session identifiers before storing them to enhance security and protect sensitive information from unauthorized access or tampering.

Rationale:

Establishing session storage is vital for secure and persistent storage of session data. Secure storage prevents leaks and unauthorized access to sensitive information like authentication tokens. Session persistence ensures seamless user interactions across requests. Additionally, scalable storage solutions, like distributed databases, enhance API scalability and performance.

Audit:

- Assess storing implementation.
- Evaluate the efficiency and suitability of storage technology.
- Evaluate and validate the effectiveness of security measures, including encryption and access controls.

Remediation:

- Choose a session storage implementation.
- Ebsure consistency across all components, services, and authentication methods of the API.
- Implement security controls around storage, including encryption and access controls.
- Monitor storage infrastructure.

2.4.6 Monitor and log session activity (Manual)

Profile Applicability:

- Level 1

Description:

Implement a procedure to monitor and log session activity.

Rationale:

Monitoring and logging session activity allows you to keep track of session activity to detect abnormal behavior, which can indicate security threats such as unauthorized access or misuse. Logging activity provides valuable data for later use in analytics and incident response, helping to identify patterns, understand user interactions, and improve security measures.

Audit:

- Review existing monitoring and logging policy.
- Evaluate the logging implementation procedure .
- Evaluate granularity - this must meet threat modeling and security requirements.
- Validate the security controls in place on log storage.

Remediation:

- Create or update the logging and monitoring policy.
- Implement a logging infrastructure to manage and store data.
- Enforce security controls around the infrastructure, put in place encryption and access control measures.

2.4.7 Audit session activity logs (Manual)

Profile Applicability:

- Level 1

Description:

Have a policy or procedure in place to review the logged data.

Rationale:

Establishing a policy for reviewing logged data provides insights into API usage, helping to understand user behavior and system interactions. This information provided helps analytics aimed at enhancing performance. A regular review of logs helps detect malicious activity patterns, which can be used for future threat modeling.

Audit:

- Review any existing logging configurations, this encompasses checking the policy, granularity and the format.
- Review the logging retention and rotation settings.
- Assess the security controls in place.
- Assess the existing inspecting procedure.

Remediation:

- Improve the logging configurations, this may include updating the policy format and granularity.
- Optimize the retention and rotation settings to ensure the log data is archived, deleted and replaced efficiently.
- Update the security controls.
- Enhance the inspecting procedure.

2.5 Third-Party Integrations

2.5.1 Identify third party dependencies (Manual)

Profile Applicability:

- Level 1

Description:

Identify third-party dependencies by determining which dependencies are being used, the developers behind them, and the code or framework they are built on.

Rationale:

Identifying third-party dependencies is crucial because they can pose a liability to system security. By assessing their security, you ensure the overall system's security integrity, mitigating potential risks associated with these dependencies.

Audit:

- Review the existing documentation.
- Review the source code to identify libraries, frameworks an so on.
- Review configuration lists, dependency management tools for identifying libraries, frameworks etc.

Remediation:

- Create an inventory of third-party software
- Regularly review and update the inventory with dependencies found in management tools, source code, configuration lists.
- Implement a solution for automating the task.

2.5.2 Identify where third-party integration take place (Manual)

Profile Applicability:

- Level 1

Description:

In the code, locate where the integration takes place.

Rationale:

Identifying where third-party integration takes place is essential as it marks potential entry points for vulnerabilities into the system. Knowing these integration points allows for thorough security assessments to ensure the integration is implemented. This helps ensure the integration is executed correctly and securely to ensure optimal functionality and mitigate potential risks. Additionally, documenting third-party integrations aids in maintaining an accurate inventory of system components and facilitates efficient troubleshooting and maintenance processes.

Audit:

- Review the documentation, check that it details the integration points.
- Review source code where third-party libraries are being imported.
- Conduct network traffic analysis to identify external connections.

Remediation:

Document all findings and any changes made.

2.5.3 Understand the integration's data flow (Manual)

Profile Applicability:

- Level 1

Description:

Be aware of how data is transferred from and to third-party systems and how third party libraries handle data through the code.

Rationale:

Understanding the data flow of integration helps identify vulnerabilities within the system, enabling proactive measures to address them. It allows verification of adherence to best practices, ensuring secure and efficient data transfer processes. Understanding data flow is essential for compliance with regulations such as GDPR, particularly concerning data transfers involving US-based systems.

Audit:

- List data flows each third-party library is performing.
- Analyze transmission protocols for encryption and security issues.
- Analyze network traffic for security controls, transmission protocols, and regulatory compliance.
- Review found practices for potential exposures and data leaks.

Remediation:

- Document flow of each third-party library.
- Conduct regular reviews and audits.

2.5.4 Identify security and privacy risks of used dependencies (Manual)

Profile Applicability:

- Level 1

Description:

Identify known and past vulnerabilities and issues associated with the any external dependencies.

Rationale:

Identifying security and privacy risks of used dependencies is crucial for several reasons. Third-party software poses a liability as it can introduce new vulnerabilities into the system. Assessing known vulnerabilities provides an overview of past and possibly present security posture, aiding in proactive risk mitigation and ensuring the integrity of the system.

Audit:

- Review the documentation and third-party dependency inventory.
- Identify and cross reference documented vulnerabilities with known databases.
- Conduct vulnerability scans or manual code review to identify vulnerabilities.
- Check dependency versions to cross-check with public databases (NVD) for known vulnerabilities (CVEs).

Remediation:

Document the findings.

2.5.5 Identify security and privacy risks within the integration (Manual)

Profile Applicability:

- Level 1

Description:

Identifying security and privacy risks within the integration involves assessing if the integration points are properly implemented leading to potential vulnerabilities and data protection issues that could arise from integrating third-party services or components.

Rationale:

Identifying security and privacy risks within integrations is vital because integrations are crucial parts of systems, and incorrectly integrated dependencies or misconfigurations can introduce data leaks.

Audit:

- Review the existing documentation.
- Find integration points and their respective findings.
- Check if findings are remediated.

Remediation:

- Review integration points from documentation.
- Conduct static code analysis and source code reviews to identify misconfigurations.
- Integrate this process in the development lifecycle.
- Document findings and remediation.

2.5.6 Identify compliance and regulatory risks of the third-party dependencies (Manual)

Profile Applicability:

- Level 1

Description:

Identify compliance and regulatory risks of the third-party dependencies, depending on the way a third-party dependency handles data, there can be regulatory risks.

Rationale:

Identifying compliance and regulatory risks associated with third-party dependencies provides an understanding of how data is being handled by these dependencies, ensuring alignment with data protection regulations and standards. Doing this provides a legal overview, helping organizations avoid penalties and legal repercussions resulting from non-compliance with regulatory requirements.

Audit:

- Identify all regulations relevant to every API.
- Assess the compliance of the third-party dependencies used.
- Evaluate impact in the case of non-compliance.
- Review the documentation for third-party dependency lists.

Remediation:

- Review the documentation for third-party dependency list
- Ensure third-party dependencies comply with relevant regulations.
- Establish contractual agreements with vendors to ensure compliance obligations and responsibilities.
- Monitor network and data flow to verify ongoing compliance.

2.5.7 Review third-party providers (Manual)

Profile Applicability:

- Level 1

Description:

Perform due diligence by assessing the third-party provider's security practices, reliability, and reputation.

Rationale:

Reviewing the third-party provider helps assess their reliability, trustworthiness, and ability to meet security and performance requirements, ensuring they align with the organization's standards.

Audit:

- Conduct security assessments before engaging with third-party providers.
- Evaluate providers through customer reviews, industry reports etc.
- Review contractual agreements, SLAs etc.
- Assess the provider's security posture through vulnerability assessments (done by you or third parties).

Remediation:

- Establish clear communication channels and procedures to address issues or concerns.
- Monitor dependencies for regulatory (non-)compliance.

2.5.8 Develop mitigation strategies for identified risks and vulnerabilities (Manual)

Profile Applicability:

- Level 1

Description:

Develop mitigation strategies for identified risks and vulnerabilities.

Rationale:

Developing mitigation strategies for identified risks and vulnerabilities enables the patching of vulnerabilities, mitigating the risk of exploitation and potential data leaks. Additionally, it helps avoid non-compliance with regulations.

Audit:

- Identify risks and vulnerabilities discovered during the audit of third-party integrations, dependencies, and providers.
- Evaluate potential impact and likelihood of exploitation for each identified vulnerability.
- Develop mitigation strategies, which may include risk acceptance, risk avoidance, risk transfer, or risk mitigation measures based on the severity and criticality of identified risks.

Remediation:

- Implement remediation measures to address identified risks and vulnerabilities, including applying patches, updates, and implementing new security controls.
- Establish incident response procedures to mitigate the impact of known or unknown security incidents.
- Monitor mitigation strategies.

2.6 System Testing and QA

2.6.1 Define the scope and the objectives of the testing (Manual)

Profile Applicability:

- Level 1

Description:

Define the scope and objectives of testing to ensure targeted and purposeful testing efforts.

Rationale:

Defining the scope and objectives of testing helps achieve goals by ensuring focused, efficient testing that aligns with specific API functions and objectives.

Audit:

- Review the scope of the test.
- Make sure the scope aligns with the goals of the test.

Remediation:

- Clarify and document scope of the testing.
- Review scope and get feedback from stakeholders.

2.6.2 Define the methodologies used in testing (Manual)

Profile Applicability:

- Level 1

Description:

Define the testing approach (black/gray/white box) based on the threat model of your API.

Rationale:

Defining the methodologies used in testing is essential as each methodology offers a different approach and perspective to evaluating the API. Different threat models require specific kinds of tests, such as gray or white box testing for insider threats, ensuring thorough coverage and effective mitigation of potential risks.

Audit:

- Review the scope of work and rules of engagement documents.
- Make sure stakeholders are aligned with decisions made.
- Ensure testing approach aligns with the threat model.

Remediation:

- Review the threat model of your API.
- Discuss approaches with stakeholders.
- Document decisions and regularly review.

2.6.3 Define the process of the system testing lifecycle (Manual)

Profile Applicability:

- Level 1

Description:

Establish a standardized testing process to be consistently used for every test conducted.

Rationale:

Defining the process of the system testing lifecycle is important as it helps stay organized by outlining the steps beforehand, making it easier to identify issues and troubleshoot problems when they arise. Additionally, it ensures structured, systematic, and repeatable testing activities from development to deployment, promoting consistency and reliability in the testing process.

Audit:

- Review the documentation.
- Assess the clarity of the lifecycle stages.
- Verify that it aligns with best practices.
- Verify that established tests undergo through the whole lifecycle.

Remediation:

- Communicate process to all stakeholders.
- Create an automation process for the lifecycle (move from dev to test env, pull test data etc).
- Establish mechanisms for continuous lifecycle enhancement.
- Document actions.

2.6.4 Set up automated testing upon code push (Manual)

Profile Applicability:

- Level 1

Description:

Have an automated testing process in place to take place upon code push

Rationale:

Setting up automated testing upon code push is essential as it provides immediate feedback on security, integration, and quality aspects, ensuring early detection of potential issues. Additionally, automated testing allows for smooth integration into the existing codebase once tests have passed, streamlining the development process and promoting efficiency.

Audit:

- Review any existing documentation.
- Verify its coverage and ensure that tests are running smoothly without errors.
- Review results of the testing.

Remediation:

- Implement automated testing frameworks and scripts.
- Integrate the tests into your CI/CD pipeline.
- Verify their effectiveness and monitor their process.
- Document all actions taken.

3 Deploy

3.1 Asset Management

3.1.1 Secure Configuration and Asset management (Manual)

Profile Applicability:

- Level 1

Description:

Ensure proper management and secure configuration of asset management and inventory.

Rationale:

Securing configuration and asset management is important as it helps to prevent leaks that could assist in malicious reconnaissance, thereby protecting sensitive information from potential attackers. Proper management also leads to a more organized and structured system, making it easier to maintain and secure.

Audit:

- Review the configuration settings for API hosts, databases, and network devices.
- Evaluate management practices.
- Verify the validity of the asset inventory.
- Check access controls and permissions.

Remediation:

- Implement a management practice and configuration for API hosts, databases, and network devices
- Maintain an asset inventory that provides a single point of truth about the org's APIs, hosts, third-party services.
- Enforce access controls and permissions to prevent leaks, adhering to the principle of least privilege.
- Periodically review and update settings and inventory.

3.1.2 Implement an Effective Release Process (Manual)

Profile Applicability:

- Level 1

Description:

Ensure the release of the API and any new versions is automated and follows a stable process.

Rationale:

Having an effective release process ensures a controlled and stable process, promoting consistency and predictability. This also clarifies the roles and responsibilities of the involved teams, enhancing coordination and efficiency.

Audit:

- Review the existing release process documentation.
- Assess the efficiency of the release process. Review every actions taken from dev to release.
- Check roles and responsibilities of each team.
- Check if the process includes all essential steps (testing, approval, deployment stages).

Remediation:

- Create documentation that standardizes the release process.
- Define and share roles among the teams involved.
- Implement feedback mechanisms.
- Document all actions.

3.1.3 Ensure the use of SSL/TLS (Manual)

Profile Applicability:

- Level 1

Description:

Verify the use of SSL/TLS for all API endpoints and versions.

Rationale:

Ensuring the use of SSL/TLS is important because it keeps data transfer and communication encrypted, protecting sensitive information from being intercepted. It also mitigates Man-in-the-Middle (MiTM) vulnerabilities, enhancing security. Additionally, using SSL/TLS helps comply with regulations that mandate secure data transmission.

Audit:

- Review SSL/TLS enforcement for all endpoints.
- Evaluate SSL/TLS versions.
- Review certification expiration dates and assess if they are compromised.

Remediation:

- Enforce SSL/TLS encryption for all API communications.
- Regularly review configurations and certificates for expiration dates and assess if they are compromised.
- Update versions when necessary.
- Monitor configurations.

3.1.4 Monitor and keep logs (Manual)

Profile Applicability:

- Level 1

Description:

Keep assets monitored and store logs in a secure place.

Rationale:

Monitoring and keeping logs are important for incident response, as they help identify what happened during a security event. Logs provide valuable insights to prevent similar issues in the future.

Audit:

- Review the existing monitoring settings.
- Ensure monitoring covers all assets.
- Ensure logging takes place and that logs are stored in a secure environment.
- Ensure storage has appropriate authorization access.

Remediation:

- Identify any assets that require monitoring.
- Enable monitoring for all identified assets with appropriate granularity.
- Store logs in a secure environment.
- Restrict log storage environments to authorized access only.
- Document all actions taken.

3.2 Use versioning

3.2.1 Use Versioning and Lifecycle Management (Manual)

Profile Applicability:

- Level 1

Description:

Versioning should be used where necessary when releasing new features.

Rationale:

Versioning when releasing new features provides a structured and stable way of releasing code updates. Additionally, it allows for easier tracking and monitoring of changes over time, facilitating better management of the development lifecycle. Versioning reduces compatibility issues, users and developers can clearly see which version of the software they are using, reducing the likelihood of compatibility issues between different versions or updates.

Audit:

- Review the version control documentation.
- Verify the existence of a versioning scheme and practice.
- Check for backward compatibility.
- Verify there are processes for removing old versions.

Remediation:

- Implement a versioning scheme (v1, v2 etc).
- Ensure backward compatibility is in place.
- Remove all outdated versions.
- Document a process for releasing code under a specific version.
- Keep track of, monitor, and log both new and old versions.

3.2.2 Ensure backward compatibility (Manual)

Profile Applicability:

- Level 1

Description:

Ensure new versions integrate seamlessly with the workflow of the previous version.

Rationale:

Ensuring backwards compatibility is important because new versions may not always be released in large updates. By ensuring that a new version of an API function is backward compatible, it can integrate with existing systems and applications when it's ready, maintaining the availability and usability of the API.

Audit:

- Ensure the process for integrating new versions works flawlessly.
- Verify the integration.

Remediation:

- QA and VAPT test on the new versions.
- Ensure the integration works without disruptions or errors and remains free of vulnerabilities.
- Document the process.

3.2.3 Remove old versions (Manual)

Profile Applicability:

- Level 1

Description:

Old versions should be removed after a period, once the majority of users have transitioned to the new version.

Rationale:

Phasing out old versions is important because they may no longer be maintained, leaving vulnerabilities unpatched and creating unnecessary attack surfaces. Additionally, outdated versions can lead to compatibility issues and pose risks to system security.

Audit:

- Review removal process of old API versions.
- Assess the automation process, if there is one in place.
- Ensure that there are no traces left once removed, such as forgotten endpoints.
- Ensure integrations from old versions with existing databases and other systems are also removed.

Remediation:

- Create an automation process that removes old versions from public and private access.
- Remove integrations from old versions that connect to existing databases and other systems.
- Document the process.

3.2.4 Notify stakeholders (Manual)

Profile Applicability:

- Level 1

Description:

All impacted parties should be notified of changes.

Rationale:

Notifying stakeholders keeps users informed about new versions including their capabilities and endpoints. Communicating how long older versions will be supported, ensures users can plan for any necessary transitions or updates.

Audit:

- Review the removal/addition procedure for API versions.
- Check if changes are currently communicated to users and stakeholders.
- Ensure communication is clear and comprehensible.

Remediation:

- Add a process for communicating API changes to the stakeholders through an appropriate communication channel.
- Ensure communication is clear and comprehensible.
- Communicate the changes and the timeline of each step.
- Document all changes made.

3.3 Create an incident response plan

3.3.1 Ensure resilience in line with RTO/RPO (Manual)

Profile Applicability:

- Level 1

Description:

Ensure that Recovery Time Objective (RTO) and Recovery Point Objective (RPO) goals are realistically achievable through established mechanisms and procedures.

Rationale:

Ensuring resilience in line with Recovery Time Objective (RTO) and Recovery Point Objective (RPO) minimizes downtime and data loss, ensuring business continuity and operational efficiency. Meeting RTO/RPO targets establishes a clear timeline striving to achieve RTO/RPO resilience enhances recovery mechanisms, recovery plans, and incident response reflexes.

Audit:

- Review resilience and disaster recovery documentation and procedures.
- Evaluate whether RTO and RPO are defined.
- Evaluate alignment resilience measures such as recovery mechanisms meet RTO and RPO goals.
- Assess the effectiveness of measures during a disaster or downtime.

Remediation:

- Define and document RTO and RPO.
- Implement resilience measures including recovery mechanisms, to meet RTO and RPO goals.
- Conduct regular resilience testing to verify effectiveness.
- Update resilience plans and procedures based on lessons learned from testing and real-world incidents.

3.3.2 Create an automated notification process (Manual)

Profile Applicability:

- Level 1

Description:

Have an automated process that notifies the relevant team when the system is down.

Rationale:

Creating an automated notification process is essential as it reduces the Recovery Time Objective (RTO), ensuring quicker response and restoration in the event of an issue, ensuring availability and improving user experience.

Audit:

- Review incident response policy.
- Ensure a process is in place for managing system downtime.
- Automate the process and ensure teams are notified through a secure channel.

Remediation:

- Create a process for system downtime, identifying necessary teams, notifying them through secure channels, and automating the process.
- Test the effectiveness of the process.
- Document all actions taken.

4 Operate

4.1 Enable monitoring and logging

4.1.1 Monitor API requests (Manual)

Profile Applicability:

- Level 1

Description:

Monitor API requests to observe real-time traffic.

Rationale:

Monitoring API requests in real-time is crucial for detecting anomalies and malicious activities through both static and dynamic analysis. It enables the identification of potential attacks and unknown vulnerabilities (zero-day exploits), enhancing security. Additionally, storing and analyzing API requests provides valuable insights into usage trends and performance, helping to optimize and secure the API infrastructure effectively.

Audit:

- Review the logging and monitoring configuration.
- Verify that the information stored is relevant and detailed.
- Review the current log retention policy.
- Ensure logs are stored securely and have s proper security controls, such as preventing unauthorized access and leaks.

Remediation:

- Set up a logging and monitoring mechanism based on the network load and available hardware (servers, load balancers etc).
- Implement log aggregation mechanisms and decide on the storage location.
- Regularly review the configuration.
- Document all actions taken.

4.1.2 Monitor API responses (Manual)

Profile Applicability:

- Level 1

Description:

Monitor API responses to observe real-time traffic.

Rationale:

Monitoring API responses is essential for detecting anomalies and vulnerabilities, such as when responses mistakenly include data from multiple users, which can indicate security breaches or coding errors. Storing and analyzing this traffic provides valuable insights into potential issues and usage patterns.

Audit:

- Review the logging and monitoring configurations.
- Verify that the information stored is relevant and detailed. Check that any sensitive information (PII, JWTs) is redacted.
- Review the log retention policy.
- Ensure logs are stored securely and have proper security controls, such as preventing unauthorized access and leaks.

Remediation:

- Set up a logging and monitoring mechanism based on the network load and available hardware (servers, load balancers etc).
- Implement a procedure to redact sensitive information such as PII from the logs.
- Implement log aggregation mechanisms and decide on the storage location.
- Regularly review the configuration.
- Document all actions taken.

4.1.3 Monitor internal API calls (Manual)

Profile Applicability:

- Level 1

Description:

Monitor calls from internal APIs and from load balancers/proxies to internal systems.

Rationale:

Monitoring internal API calls helps detect malicious insiders and Advanced Persistent Threats (APTs) that might be operating within the network. It also allows for the identification of second-order attacks, such as HTTP smuggling, which can exploit vulnerabilities in the interaction between different systems and services.

Audit:

- Review the current logging and monitoring configuration.
- Verify the internal APIs and systems are also included in the monitoring process.
- Review the log retention policy.
- Ensure logs are stored securely and have proper security controls, such as preventing unauthorized access and leaks.

Remediation:

- Identify internal APIs and their exit/entry points to the Internet (proxy, load balancer).
- Set a logging and monitoring mechanism based on the network load and available hardware (servers, load balancers etc).
- Implement log aggregation mechanisms and decide on the storage location.
- Regularly review the configuration.
- Document all actions taken.

4.2 Maintain system availability

4.2.1 Monitor Availability and Performance (Manual)

Profile Applicability:

- Level 1

Description:

Monitor availability and performance by tracking system metrics such as CPU usage, RAM, network latency, and workload, as well as monitoring system uptime to detect and address downtime issues.

Rationale:

Monitoring availability and performance provides an overview of the system workload and performance, offering insights into areas for enhancement, as well as identifying issues that require fixing or may result in downtime, ensuring proactive management of system health and efficiency.

Audit:

- Evaluate the existing monitoring tools and systems.
- Review performance thresholds and alerting mechanisms for timely detection.
- Validate that monitoring covers key aspects (resource utilization, network latency).
- Assess the frequency and granularity of monitoring checks.

Remediation:

- Configure monitoring mechanisms to cover system aspects of the API systems.
- Develop performance baselines which should serve as a "bare minimum".
- Configure alerting mechanisms.
- Conduct regular monitoring mechanisms reviews.
- Document all actions taken.

4.2.2 Ensure scalability under load (Manual)

Profile Applicability:

- Level 1

Description:

The ability of the API and its components to scale as user traffic demands.

Rationale:

Ensuring scalability under load is crucial to maintaining API availability and performance, even during high usage spikes. This prevents performance issues, ensuring a smooth user experience and maintaining business continuity. By avoiding downtimes and lag, customer satisfaction remains high, which is essential for the business's reputation and reliability. Scalability supports the seamless handling of increased demand, securing long-term operational stability and success.

Audit:

- Evaluate the existing scalability testing procedures.
- Review infrastructure and resource allocation when scaling.
- Ensure scaling is an automated process.

Remediation:

- Perform load and stress testing to assess the API's traffic-handling capacity.
- Modify infrastructure and allocate resources accordingly to accommodate scaling requirements.
- Automate scaling processes to streamline efficiency and responsiveness.
- Continuously monitor the scaling process for ongoing optimization.

4.3 Ensure compliance and testing

4.3.1 Audit requirements and reporting (Manual)

Profile Applicability:

- Level 1

Description:

Audits are essential to ensure that the security posture and compliance measures are adequate.

Rationale:

Considering audits is essential because it demonstrates adherence to regulatory requirements, industry standards, and internal policies. It also enables timely detection and response to security incidents, breaches, and compliance violations, helping to mitigate risks and prevent future incidents.

Audit:

- View the audit logs and reports to verify compliance.
- Validate audit events are logged accurately, including user actions, system events, and security incidents.
- Assess the completeness and accuracy of audit trails.

Remediation:

- Enhance audit logging mechanisms for complete coverage of the system.
- Automate the audit report generation.
- Conduct regular reviews of audit logs to identify anomalies, trends, and for optimization.

4.3.2 Align Data Retention with Compliance Standards (Manual)

Profile Applicability:

- Level 1

Description:

Data retention and storage must comply with regulations and industry standards for best performance and security.

Rationale:

Aligning Data Retention with Compliance Standards is crucial because it ensures that data is retained according to regulatory requirements, thus avoiding legal risks. Additionally, it aids in maintaining security and privacy standards, while enhancing transparency, accountability, and trustworthiness in data handling practices.

Audit:

- Review data retention policies and procedures to ensure compliance.
- Verify data retention periods are defined based on legal requirements.
- Assess data storage and practices ensure secure data retention.
- Validate deletion process is in place for data that exceed retention periods.

Remediation:

- Revise data retention policies and procedures to align with current regulatory requirements and industry best practices.
- Classify data that requires retention.
- Implement an automated process for the retention process.
- Conduct periodic audits for continuous compliance and optimization.

4.3.3 Regular VAPT for Enhanced Security (Manual)

Profile Applicability:

- Level 1

Description:

Regular Vulnerability Assessment and Penetration Testing (VAPT) is conducted to enhance security by identifying and addressing potential vulnerabilities and threats.

Rationale:

Regular VAPT is crucial as it mitigates risks by catching vulnerabilities early, assesses an attacker's ability to enumerate the API in black box scenarios, uncovers system weaknesses beyond vulnerabilities, and fulfills regulatory requirements.

Audit:

- Review vulnerability assessment and penetration testing procedures and schedules, incorporating legal documents and establishing clear communication channels.
- Validate that VAPT activities cover all aspects of the API infrastructure.
- Ensure that VAPT findings are documented, prioritized, and addressed promptly.

Remediation:

- Schedule regular VAPT activities based on risk assessments, compliance requirements, and industry best practices.
- Ensure tests encompass a wide range of scenarios, include various vulnerability tests, and incorporate both automated and manual testing methods.
- Establish a procedure for prioritizing and remediating vulnerabilities.

5 Decommission

5.1 Establish a Decommissioning Plan

5.1.1 Determine the Required Actions (Manual)

Profile Applicability:

- Level 1

Description:

Identify what should be decommissioned.

Rationale:

Identifying what should be decommissioned is essential as it provides a comprehensive overview of the project's timeline and tasks, aiding in better organization. This process ensures clarity in planning and facilitates effective communication with stakeholders by articulating the specific elements that will be phased out. By delineating decommissioning objectives early on, teams can streamline their efforts, allocate resources efficiently, and mitigate potential disruptions to ongoing operations.

Audit:

- Examine the decommissioning plan to assess the timelines and tasks outlined.
- Ensure the plan is shared and communicated to all teams involved.
- Establish communication with stakeholders regarding the decommissioning process.

Remediation:

- Organize what tasks have to be completed by the end of the project.
- Determine suitable timelines for each task in collaboration with the responsible team.
- Verify that documentation of actions is being conducted.

5.1.2 Establish a data migration strategy (Manual)

Profile Applicability:

- Level 1

Description:

Create a migration plan that determines whether data should be migrated, erased, or archived.

Rationale:

Planning the migration ensures that data, which typically needs to be migrated and stored, is managed securely. This process mitigates the risks of leaks or data loss by implementing organized and controlled transfer procedures.

Audit:

- Examine the data migration plan to assess the timelines and tasks outlined.
- Ensure the plan is shared and communicated to all teams involved.
- Check that the process is automated.

Remediation:

- Organize what tasks have to be completed by the end of the migration.
- Assign tasks to each team involved.
- Automate the process and test its performance.
- Determine a suitable timeline.
- Verify that documentation of actions is being conducted.

5.1.3 Decide on a timeline (Manual)

Profile Applicability:

- Level 1

Description:

Create a timeline for decommissioning outlining the tasks and responsibilities for each team involved.

Rationale:

Creating a timeline for decommissioning, detailing tasks and responsibilities for each team involved, offers a clear plan of action, minimizing miscommunications between interdependent teams. This ensures coordination and execution of decommissioning activities, promoting efficiency and avoiding delays or disruptions in the process.

Audit:

- Review the existing action plan.
- Determine that suitable timelines are in place.
- Collect feedback from the teams regarding the feasibility of the proposed timeline.

Remediation:

- Put in place, or update the action plan.
- Communicate with the teams on the timeline needed for actions.
- Inform the teams about the timeline required for each action.
- Document timeline and actions.

5.2 Communicate the Decommission Plan with Stakeholders

5.2.1 Notify users (Manual)

Profile Applicability:

- Level 1

Description:

Provide all stakeholders with clear timelines, propose alternative solutions, and outline the actions to be taken.

Rationale:

Clear communication helps avoid errors, bad UX, and security issues.

Audit:

- Ensure communication channels are set.
- Ensure the communication with stakeholders encompasses the action plan, future steps, and any necessary user actions.
- Ensure the communication with stakeholders is clear and comprehensible.

Remediation:

- Chose a reliable communication channel.
- Notify stakeholders about the plan and inform them of any actions they should take.
- Notify them of what actions they may / may not need to take.
- Document all actions taken.

5.2.2 Notify third-party companies (Manual)

Profile Applicability:

- Level 1

Description:

Notify third-party companies utilizing your API or whose APIs/libraries/systems your API integrates with regarding the decommissioning process, providing clear timelines, alternative solutions, and outlining the actions that will be taken.

Rationale:

Clear communication is essential as it provides third parties with insight into your actions, ensuring transparency throughout the decommissioning process. Notifying them allows them to take necessary actions, such as terminating contracts with you or removing systems supporting your integration, thereby facilitating a smooth transition and minimizing disruptions for all parties involved.

Audit:

- Ensure communication channels are in place.
- Ensure communication includes the action plan, future actions, and actions that may need to be taken.
- Ensure the message is clear and comprehensible.

Remediation:

- Choose a reliable communication channel if one is not in place.
- Notify stakeholders about the plan and inform them of any actions they should take.
- Communicate in a clear and comprehensible way
- Document actions.

5.2.3 Notify developers (Manual)

Profile Applicability:

- Level 1

Description:

Provide developers with transparent timelines, propose alternative solutions, and detail the actions to be executed.

Rationale:

Clear communication helps in mitigating organizational errors and misunderstandings, while also providing developers with a better understanding of the systems.

Audit:

- Ensure communication channels are in place.
- Ensure communication includes the action plan, future actions, and developer actions that may need to be taken.
- Ensure the message is clear and comprehensible.

Remediation:

- Choose a reliable communication channel if one is not in place.
- Notify stakeholders about the plan and inform them of any actions they should take.
- Communicate in a clear and comprehensible way
- Document all actions taken.

5.3 Apply Actions

5.3.1 Terminate services (Manual)

Profile Applicability:

- Level 1

Description:

Cease services as part of the API decommissioning process.

Rationale:

Ceasing services as part of the API decommissioning process ensures a smooth transition. This allows stakeholders time to adjust to the changes, minimizing interruptions and errors. By coordinating the cessation of services effectively, the decommissioning process becomes seamless, maintaining operational continuity and reducing the risk of disruptions.

Audit:

- Review the existing service termination plan.
- Ensure the plan is realistic and that all teams have tasks that can be completed on time.
- Review monitoring logs for best time to terminate.
- Ensure proper handling of termination with stakeholders and help users identify alternatives to the terminated service.
- Verify the termination.

Remediation:

- Put a termination plan in place. Include steps needed for complete API service termination.
- Plan out a schedule and assign tasks to each team.
- Optimize your strategies to minimize the impact on the service, including fine-tuned error handling and directing users to the correct API.
- Document all actions taken.

5.3.2 Retire infrastructure (Manual)

Profile Applicability:

- Level 1

Description:

Retire any physical or cloud infrastructure that was supporting the service that was terminated.

Rationale:

Retiring any physical or cloud infrastructure that supported the terminated service is imperative for several reasons. Timely retirement minimizes unnecessary costs associated with maintaining unused infrastructure. It also reduces the attack surface, as lingering infrastructure poses potential security risks. Additionally, it ensures compliance with regulations, particularly concerning non-retired infrastructure containing personally identifiable information (PII), thereby mitigating legal and reputational risks.

Audit:

- Review the existing infrastructure retirement plan.
- Check that the plan and schedule is realistic and all teams have tasks that can be delivered on time
- Validate that data is stored securely or erased.
- Ensure hardware is securely decommissioned and follows compliance and industry standards.

Remediation:

- Schedule a plan for retiring hardware.
- Share tasks to teams that are involved in decommissioning infrastructure.
- Perform data sanitization and securely erase or store data, according to regulations.
- Verify the plan's smooth execution.
- Automate the process.
- Document all actions taken.

5.3.3 Review network configurations (Manual)

Profile Applicability:

- Level 1

Description:

Review existing network configuration before termination of services and infrastructure.

Rationale:

Forgotten network configurations can introduce vulnerabilities that malicious actors could exploit, posing security risks to the network. For instance, configurations such as referenced subdomains that are no longer tied to infrastructure or forgotten CNAME records can lead to subdomain takeovers, potentially compromising the integrity of the network and its assets.

Audit:

- Review the infrastructure documentation.
- Review the configuration and network that is associated with its hardwre.
- Make sure there are no dangling settings and network configurations, such as:
    - DNS records are pulled down.
    - Any referenced (sub-)domains are changed/deleted.
    - CNAME and DNS records that point to expired CMS domains are pulled.

Remediation:

- Put correct configurations in place:
    - Pull down any unnecessary DNS records.
    - Any referenced (sub-)domains to other sites and/or APIs should be changed/deleted.
    - CNAME and DNS records that point to expired CMS domains should be deleted to avoid subdomain takeovers.

Appendix: Summary Table

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1	**Design**		
1.1	**Documentation**		
1.1.1	Define a Format (Manual)	☐	☐
1.1.2	Specify Endpoints (Manual)	☐	☐
1.1.3	Specify Methods (Manual)	☐	☐
1.1.4	Specify Data Models (Manual)	☐	☐
1.1.5	Select an Automation-Friendly Format (Manual)	☐	☐
1.1.6	Specify Endpoint Parameters (Manual)	☐	☐
1.1.7	Make Documentation the Single Source of Truth (Manual)	☐	☐
1.1.8	Establish Clear and Comprehensive Documentation Standards (Manual)	☐	☐
1.1.9	Communicate Documentation Updates Across all Teams (Manual)	☐	☐
1.2	**Authentication and Authorization**		
1.2.1	Define Authentication and Authorization Requirements (Manual)	☐	☐
1.2.2	Determine API Authorization Approach (Manual)	☐	☐
1.2.3	Define User Roles (Manual)	☐	☐
1.2.4	Decide on role-based access controls (Manual)	☐	☐
1.2.5	Choose Authentication Mechanisms (Manual)	☐	☐
1.3	**Data Security and Privacy**		

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
1.3.1	Assess Data Protection Needs (Manual)	☐	☐
1.3.2	Determine Applicable Compliance Standards (Manual)	☐	☐
2	**Develop**		
2.1	**Identity and Access Management**		
2.1.1	Define User Identity Management Protocols within the System (Manual)	☐	☐
2.1.2	Define User Types and Required Information (Manual)	☐	☐
2.1.3	Define Authentication Methods and Protocols (Manual)	☐	☐
2.1.4	Define Access Levels for Each Role (Manual)	☐	☐
2.1.5	Define Authorization Models to Enforce Access Restrictions (Manual)	☐	☐
2.1.6	Implement Cryptographic Techniques to Secure User Credentials and Authentication Tokens (Manual)	☐	☐
2.2	**API Endpoint Security**		
2.2.1	Identify the security requirements for each API endpoint (Manual)	☐	☐
2.2.2	Enforce TLS/SSL protocols (Manual)	☐	☐
2.2.3	Enforce input validation and sanitization (Manual)	☐	☐
2.2.4	Enforce Rate Limiting mechanisms (Manual)	☐	☐
2.2.5	Define logging and monitoring procedures (Manual)	☐	☐
2.2.6	Establish Practices Based on Regulations and Industry Standards (Manual)	☐	☐
2.3	**Use Proper Error Handling**		
2.3.1	Identify possible error occurrences (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
2.3.2	Establish standardized error handling procedures (Manual)	☐	☐
2.3.3	Enforce error logging (Manual)	☐	☐
2.3.4	Enforce error monitoring (Manual)	☐	☐
2.3.5	Establish a strategy for error recovery (Manual)	☐	☐
2.4	**Session Management**		
2.4.1	Define the session lifecycle stages (Manual)	☐	☐
2.4.2	Define how a session is initialized (Manual)	☐	☐
2.4.3	Define session duration and validity (Manual)	☐	☐
2.4.4	Set up mechanisms to maintain sessions (Manual)	☐	☐
2.4.5	Define session storage procedures (Manual)	☐	☐
2.4.6	Monitor and log session activity (Manual)	☐	☐
2.4.7	Audit session activity logs (Manual)	☐	☐
2.5	**Third-Party Integrations**		
2.5.1	Identify third party dependencies (Manual)	☐	☐
2.5.2	Identify where third-party integration take place (Manual)	☐	☐
2.5.3	Understand the integration’s data flow (Manual)	☐	☐
2.5.4	Identify security and privacy risks of used dependencies (Manual)	☐	☐
2.5.5	Identify security and privacy risks within the integration (Manual)	☐	☐
2.5.6	Identify compliance and regulatory risks of the third-party dependencies (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
2.5.7	Review third-party providers (Manual)	☐	☐
2.5.8	Develop mitigation strategies for identified risks and vulnerabilities (Manual)	☐	☐
2.6	**System Testing and QA**		
2.6.1	Define the scope and the objectives of the testing (Manual)	☐	☐
2.6.2	Define the methodologies used in testing (Manual)	☐	☐
2.6.3	Define the process of the system testing lifecycle (Manual)	☐	☐
2.6.4	Set up automated testing upon code push (Manual)	☐	☐
3	**Deploy**		
3.1	**Asset Management**		
3.1.1	Secure Configuration and Asset management (Manual)	☐	☐
3.1.2	Implement an Effective Release Process (Manual)	☐	☐
3.1.3	Ensure the use of SSL/TLS (Manual)	☐	☐
3.1.4	Monitor and keep logs (Manual)	☐	☐
3.2	**Use versioning**		
3.2.1	Use Versioning and Lifecycle Management (Manual)	☐	☐
3.2.2	Ensure backward compatibility (Manual)	☐	☐
3.2.3	Remove old versions (Manual)	☐	☐
3.2.4	Notify stakeholders (Manual)	☐	☐
3.3	**Create an incident response plan**		
3.3.1	Ensure resilience in line with RTO/RPO (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
3.3.2	Create an automated notification process (Manual)	☐	☐
4	**Operate**		
4.1	**Enable monitoring and logging**		
4.1.1	Monitor API requests (Manual)	☐	☐
4.1.2	Monitor API responses (Manual)	☐	☐
4.1.3	Monitor internal API calls (Manual)	☐	☐
4.2	**Maintain system availability**		
4.2.1	Monitor Availability and Performance (Manual)	☐	☐
4.2.2	Ensure scalability under load (Manual)	☐	☐
4.3	**Ensure compliance and testing**		
4.3.1	Audit requirements and reporting (Manual)	☐	☐
4.3.2	Align Data Retention with Compliance Standards (Manual)	☐	☐
4.3.3	Regular VAPT for Enhanced Security (Manual)	☐	☐
5	**Decommission**		
5.1	**Establish a Decommissioning Plan**		
5.1.1	Determine the Required Actions (Manual)	☐	☐
5.1.2	Establish a data migration strategy (Manual)	☐	☐
5.1.3	Decide on a timeline (Manual)	☐	☐
5.2	**Communicate the Decommission Plan with Stakeholders**		
5.2.1	Notify users (Manual)	☐	☐
5.2.2	Notify third-party companies (Manual)	☐	☐

CIS Benchmark Recommendation		Set Correctly	
		Yes	No
5.2.3	Notify developers (Manual)	☐	☐
5.3	**Apply Actions**		
5.3.1	Terminate services (Manual)	☐	☐
5.3.2	Retire infrastructure (Manual)	☐	☐
5.3.3	Review network configurations (Manual)	☐	☐