APPLICATION PROGRAMMING INTERFACE (API) SECURITY REQUIREMENTS GUIDE OVERVIEW

Version 1, Release 1

11 September 2025

Developed by DISA for the DOD

Trademark Information

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our users, and do not constitute or imply endorsement by the Defense Information Systems Agency (DISA) of any nonfederal entity, event, product, service, or enterprise.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

1. INTRODUCTION

1.1 Executive Summary

The Application Programming Interface (API) Security Requirements Guide (SRG) provides the technical security policies and requirements for applying security concepts to systems.

The API SRG outlines essential security controls and best practices to protect APIs from common threats such as unauthorized access, data breaches, and injection attacks. This guide covers key areas including authentication, authorization, encryption, rate limiting, input validation, logging, and secure development practices. This guide helps ensure that APIs are developed and maintained with security in mind, reducing risks and supporting safe integration and communication between systems.

1.1.1 Security Requirements Guides (SRGs)

Security Requirements Guides are collections of requirements applicable to a given technology family. They represent an intermediate step between Control Correlation Identifiers (CCIs) and Security Technical Implementation Guides (STIGs). CCIs represent discrete, measurable, and actionable items sourced from Information Assurance (IA) controls defined in a policy, such as the National Institute of Standards and Technology (NIST) Special Publication (SP) 800-53. STIGs provide product-specific information for validating and attaining compliance with requirements defined in the SRG for that product's technology area.

There are four core SRGs: Application, Network, Operating System, and Policy. Each addresses the applicable CCIs in the context of the technology family. Subordinate to the core SRGs, Technology SRGs are developed to address the technologies at a more granular level.

This API SRG is based on the Application SRG. The API SRG contains general check and fix information that can be used for products for which STIGs do not exist.

The STIGs based on this SRG will provide the product-specific technical implementation guidance for that product. The STIG will contain the specific check and fix information for the product it covers.

SRG Hierarchy example:

Application SRG
|__Database SRG
|__Microsoft SQL Server 2016 STIG

The SRG relationship and structure provides the ability to identify requirements that may be considered not applicable for a given technology family and to provide appropriate justification. It also provides the structure to identify variations in specific values based on the technology family. These variations will be captured once and will propagate down to the Technology SRGs and then to the STIGs. This will eliminate the need for each product-specific STIG to address items that are not applicable.

1.1.2 SRG Naming Standards

To establish consistency across the SRGs, a naming standard for the Group Title and STIGIDs has been established.

Technology SRG Naming Standards

For Technology SRG Group Title and STIGIDs, the following applies:

{Core SRG value}-{Technology SRG}-{5- or 6-digit numeric sequence number}

Examples:

> *SRG-NET-000001-RTR-000001*
> *SRG-APP-000001-COL-000001*
> *SRG-NET-000001-VVSM-00001*
> *SRG-OS-000001-UNIX-000001*

Checks/fixes will be included at this level in a general form. These checks and fixes will apply for any STIGs that are created for products that do not have product-specific check and fix guidance.

1.2 Authority

Department of Defense Instruction (DODI) 8500.01 requires that "all IT [information technology] that receives, processes, stores, displays, or transmits DOD information will be […] configured […] consistent with applicable DOD cybersecurity policies, standards, and architectures." The instruction tasks that DISA "develops and maintains control correlation identifiers (CCIs), security requirements guides (SRGs), security technical implementation guides (STIGs), and mobile code risk categories and usage guides that implement and are consistent with DOD cybersecurity policies, standards, architectures, security controls, and validation procedures, with the support of the NSA/CSS [National Security Agency/Central Security Service], using input from stakeholders, and using automation whenever possible." This document is provided under the authority of DODI 8500.01.

Although the use of the principles and guidelines in these SRGs/STIGs provides an environment that contributes to the security requirements of DOD systems, applicable NIST SP 800-53 cybersecurity controls must be applied to all systems and architectures based on the Committee on National Security Systems (CNSS) Instruction (CNSSI) 1253.

1.2.1 Relationship to STIGs

The SRG defines the requirements for various technology families, and the STIGs are the technical implementation guidelines for specific products. A single SRG/STIG is not all-inclusive for a given system, which may include but is not limited to Database, Web Server, and Domain Name System (DNS) SRGs/STIGs. For a given system, compliance with all (multiple) SRGs/STIGs applicable to a system is required.

1.3   Vulnerability Severity Category Code Definitions

Severity Category Codes (referred to as CAT) are a measure of vulnerabilities used to assess a facility or system security posture. Each security policy specified in this document is assigned a Severity Category Code of CAT I, II, or III.

Table 1-1: Vulnerability Severity Category Code Definitions

Category	DISA Category Code Guidelines
CAT I	Any vulnerability, the exploitation of which will **directly and immediately** result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Availability, or Integrity.

1.3.1   SRG and STIG Distribution

Parties within the DOD and federal government's computing environments can obtain the applicable STIG from the DOD Cyber Exchange website at https://cyber.mil/. This site contains the latest copies of STIGs, SRGs, and other related security information. Those without a Common Access Card (CAC) that has DOD Certificates can obtain the STIG from https://public.cyber.mil/.

1.4   Document Revisions

Comments or proposed revisions to this document should be sent via email to the following address: disa.stig_spt@mail.mil. DISA will coordinate all change requests with the relevant DOD organizations before inclusion in this document. Approved changes will be made in accordance with the DISA maintenance release schedule.

1.5   Other Considerations

DISA accepts no liability for the consequences of applying specific configuration settings made on the basis of the SRGs/STIGs. It must be noted that the configuration settings specified should be evaluated in a local, representative test environment before implementation in a production environment, especially within large user populations. The extensive variety of environments makes it impossible to test these configuration settings for all potential software configurations.

For some production environments, failure to test before implementation may lead to a loss of required functionality. Evaluating the risks and benefits to a system's particular circumstances and requirements is the system owner's responsibility. The evaluated risks resulting from not applying specified configuration settings must be approved by the responsible AO. Furthermore, DISA implies no warranty that the application of all specified configurations will make a system 100 percent secure.

Security guidance is provided for the DOD. While other agencies and organizations are free to use it, care must be given to ensure that all applicable security guidance is applied at both the device hardening level and the architectural level due to the fact that some settings may not be configurable in environments outside the DOD architecture.

1.6   Product Approval Disclaimer

The existence of a STIG does not equate to DOD approval for the procurement or use of a product.

STIGs provide configurable operational security guidance for products being used by the DOD. STIGs, along with vendor confidential documentation, also provide a basis for assessing compliance with cybersecurity controls/control enhancements, which supports system assessment and authorization (A&A) under the DOD Risk Management Framework (RMF). Department of Defense AOs may request available vendor confidential documentation for a product that has a STIG for product evaluation and RMF purposes from disa.stig_spt@mail.mil. This documentation is not published for general access to protect the vendor's proprietary information.

AOs have the purview to determine product use/approval in accordance with (IAW) DOD policy and through RMF risk acceptance. Inputs into acquisition or pre-acquisition product selection include such processes as:

- National Information Assurance Partnership (NIAP) evaluation for National Security Systems (NSS) (https://www.niap-ccevs.org/) IAW CNSSP #11.
- National Institute of Standards and Technology (NIST) Cryptographic Module Validation Program (CMVP) (https://csrc.nist.gov/groups/STM/cmvp/) IAW federal/DOD mandated standards.

2. ASSESSMENT CONSIDERATIONS

2.1 NIST SP 800-53 Requirements

All applicable baseline technical NIST SP 800-53 requirements and security best practice requirements are included in this SRG.

CNSSI 1253 defines the required controls for DOD systems based on confidentiality, integrity, and availability (baseline) of the given information system. In all cases, CNSSI 1253, along with required baselines, will serve as the policy requirement for any given asset or information system.

2.2 General Procedures

This SRG has procedures intended to provide appropriate evaluation and remediation functions for a typically configured system. These procedures are not product-specific and are intended for use when a product-specific STIG is not available.

3. CONCEPTS AND TERMINOLOGY CONVENTIONS

- Application Programming Interface: An interface that allows different parts of a system, often web based, to interact and exchange information.
- Client: The system or application that initiates requests to the API.
- Resource: A specific URL path within an API that performs a defined function.
- Access token: A time-limited credential (often a JSON web token) used by a client to access protected API resources after successful authentication.
- Encryption: The process of encoding data to protect it during transmission or storage.
- Input validation: The process of ensuring incoming data is correct, safe, and expected before processing.
- Nonrepudiation: The assurance that actions taken via the API can be definitively traced to a specific user or system, typically supported by strong authentication and logging.
- Rate limiting/throttling: Controlling the number of API requests from clients over a time period to prevent abuse such as denial-of-service (DoS) attacks.
- Logging and auditing: Capturing detailed records of API activity to enable incident detection, investigation, and compliance monitoring.
- Least privilege: A principle requiring that users, systems, and services have only the minimum access necessary to perform their functions.

3.1 API Architecture

APIs serve as the structured interface through which different software components, services, or systems communicate. They are a central element in modern software architecture, especially in distributed systems, microservices, and cloud-native environments.

3.1.1 Layered Architecture Integration

- APIs act as the middle layer in a multi-tier architecture.
  - Frontend (Client): Web or mobile applications that consume APIs to display data or perform actions.
  - API layer: Handles requests, processes business logic, enforces security, and interacts with backend services.
  - Backend services and databases: Store and manage data or perform heavy processing; exposed via APIs.

3.1.2 Architectural Styles

- Representational State Transfer (REST): Uses HTTP methods (GET, POST, etc.), emphasizes stateless communication, and works with resource-based URLs.
- GraphQL: Allows clients to query precisely the data they need, reducing over-fetching and under-fetching.

- gRPC: A high-performance Remote Procedure Call (RPC) framework using HTTP/2 and protocol buffers for communication; suitable for internal microservice calls.

3.1.3   Microservices Enablement

APIs are foundational in microservices architecture, where each service exposes functionality via an API. This allows services to be:

- Independently deployable.
- Language agnostic.
- Scalable based on demand.

3.1.4   API Gateway

API Gateway is any combination of reverse proxies and request processing units that implement organizational policies for client requests to a resource. The component(s) must implement features as follows:

- Routing and load balancing.
- Redirect/refresh unauthenticated requests using a DOD ICAM.
- Enforce authorization using enterprise security profiles/policies.
- Rate limiting and throttling.
- Request and response transformation.
- Monitoring and logging.

3.1.5   Security Architecture

APIs must be secured at every layer. Mechanisms include:

- OAuth 2.0/OpenID Connect for delegated access.
- TLS (HTTPS) for encrypted communication.
- API keys or JWTs for authentication and session control.
- WAFs and security policies to detect and block malicious traffic.

Figure 3-1: API Architecture
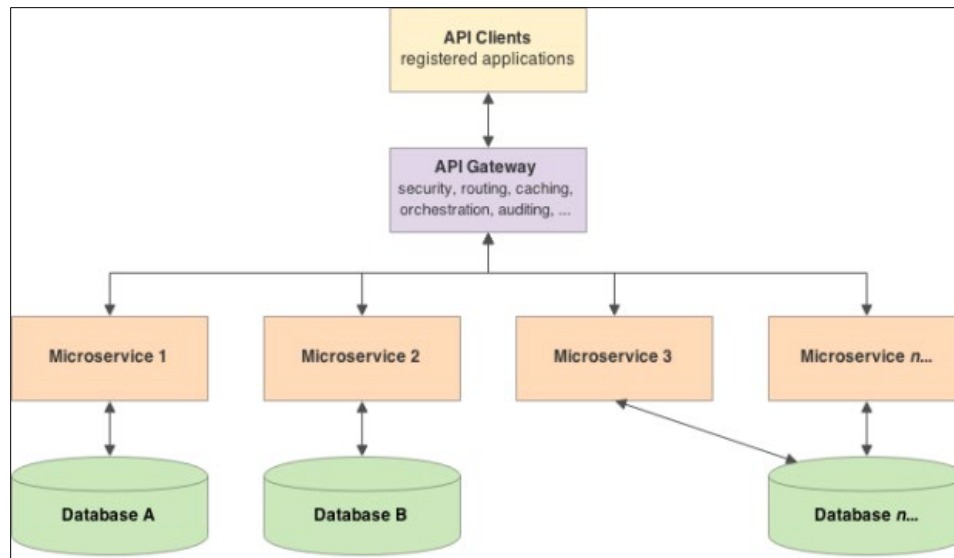
4. GENERAL SECURITY REQUIREMENTS

APIs must be secured through a combination of authentication, authorization, encryption, and input validation to protect against unauthorized access and data breaches. Strong authentication mechanisms such as OAuth 2.0 or API keys must be enforced, along with role- or attribute-based access controls. Input must be strictly validated and output properly encoded to prevent injection attacks.

APIs must implement rate limiting and throttling to guard against abuse and DoS attacks. Sensitive data must be protected both in transit and at rest, with care taken not to expose it in logs or error messages. Logging and monitoring must be in place to detect and respond to suspicious activity. Error handling must avoid revealing internal system details.

Security testing—including code reviews, vulnerability scanning, and penetration testing—must be conducted regularly. Adopting the principle of least privilege and documenting security expectations and disclosure processes further strengthens the overall API security posture.