



# APPLICATION SERVER SECURITY REQUIREMENTS GUIDE (SRG) TECHNOLOGY OVERVIEW

## Version 2, Release 2

## 23 October 2015

## Developed by DISA for the DoD

## Trademark Information

# TABLE OF CONTENTS

**Page**

# LIST OF TABLES

**Page**

# 1. INTRODUCTION

## 1.1 Executive Summary

This Application Server Security Requirements Guide (SRG) addresses the software framework used to create an application-server implementation, without regard to what the application functions are. The framework is comprised of services such as data connection services, security, transaction support, load balancing, management, and APIs to extend the application server. Specific software technologies used within the application server, e.g., web server, database, auditing system, etc., must meet the requirements for the specific technology SRG and/or STIG.

### 1.1.1 Security Requirements Guides (SRGs)

SRGs are collections of requirements applicable to a given technology family. SRGs 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, there are Technology SRGs developed to address the technologies at a more granular level.

This Application Server SRG is based on the Application SRG. This Application Server SRG contains general check and fix information that can be utilized 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*
*|__MS SQL Server 2005 STIG*

The SRG relationship and structure provides the ability to identify requirements that may be considered not applicable for a given technology family and 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

In an effort 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

DoD Instruction (DoDI) 8500.01 requires that "all IT that receives, processes, stores, displays, or transmits DoD information will be […] configured […] consistent with applicable DoD cybersecurity policies, standards, and architectures" and tasks that Defense Information Systems Agency (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, 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 provide an environment that contributes to the security requirements of DoD systems, applicable NIST SP 800-53 cybersecurity controls need to 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**

| | **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. |

## 1.4 SRG and STIG Distribution

Parties within the DoD and Federal Government's computing environments can obtain the applicable STIG from the Information Assurance Support Environment (IASE) website. This site contains the latest copies of any STIGs, SRGs, and other related security information. The address for the IASE site is http://iase.disa.mil/.

## 1.5 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.

## 1.6 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 configurations 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 Authorizing

Official. Furthermore, DISA implies no warranty that the application of all specified configurations will make a system 100% secure.

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

# 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 that are 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

An application server is a platform used for providing a runtime environment for applications. The runtime environment is comprised of services such as data connection services, security, transaction support, load balancing, and management. Although there is an array of application servers supporting different platforms, the majority is web server-oriented and is designed for running Java-based applications. In addition to providing the aforementioned services, application servers also provide Application Programming Interfaces (APIs) that developers can utilize in their programming to extend and enhance the features and functionality of their applications without having to develop additional functionality themselves. These APIs are language specific and are based upon the language the application server is designed to support. From a Java application server perspective, a Java 2 Enterprise Edition (J2EE) server is derived from, and provides additional features and functionality over, the Java Standard Edition (JSE) application server. Some of the additional features provided by a J2EE application server include, but are not limited to, JDBC connectors, messaging services, and web services. This SRG is written so it may be applied to all application servers.

Many application server products are offered as open source solutions. In some instances, a vendor will provide an open source application server that provides limited functionality and support while simultaneously offering a commercial product based on the same code but with additional features, functionality, and support.

## 3.1 Components

An application server product will consist primarily of the following core software-based components:

- An HTTPD server for providing web-based client access to application users.
- Java application servers utilizing “Containers”, which provide application components one or multiple runtime environments as well as security and transaction management services.
- A Management interface for configuring application server components. This is typically a web-based interface.
- Load balancers for balancing application traffic within the cluster nodes. These are most often HTTPD or software based.
- Application Programmable Interfaces (APIs). APIs are used to provide developers with additional features they can call from within their programs.

Although other functionality exists and will vary according to vendor and application server language and purpose, these are the core application server components shared by the majority of application servers.

## 3.2 Protocols

An application server will include and be capable of utilizing a wide array of protocols in the furtherance of its mission. These protocols include, but are not limited to:

- HTTP - Hyper Text Transport Protocol
- HTTPS - Hyper Text Transport Protocol over SSL
- SSL - Secure Sockets Layer
- XML - Extensible Markup Language
- HTML - Hyper Text Markup Language
- RMI - Remote Method Invocation
- IIOP - Internet Inter-Orb Protocol
- SOAP - Simple Object Access Protocol

In addition to the aforementioned protocols, there are additional protocols used for Java and C language application development purposes. Protocols related to application development aspects of the application server are not in scope and will not be addressed within this guidance.

## 3.3 Features and Functionality

An application server will provide features and functionality for managing and deploying applications. All products will vary in terms of how features are presented; however, most all will include remote web-based management as well as command line-driven administrative management capabilities, clustering for application redundancy, load balancing, and failover as well as session management, thread pool, and connection pools for database connectivity.

Additionally, many application servers provide extensible management APIs that can be used to extend and enhance the management features offered out of the box by the vendor. In the case of Java-based application servers, these are referred to as Java Management Extensions (JMX) interfaces. Changes to management features via JMX are not directly addressed here, as this is considered a custom-developed extension of an existing management capability. Therefore, the developer of those enhancements will need to ensure that any modifications they make regarding JRE server management capability complies with existing application development policy as stated in the Application Security and Development STIG, which is available on IASE.

Another functionality offered by most application servers is the ability to integrate with Software Development Kits (SDK) used by developers to create software applications. Many SDKs offer the capability to upload or deploy application code directly to the application server. This is done to ease development efforts and increase productivity, but this can also create security risks.

While allowing developers the ability to deploy their own applications may be an acceptable practice in a test environment, care needs to be taken to ensure that developers do not have direct upload access to the production environment. Rather, a formal application test and

deployment process needs to be utilized in order to ensure any changes made to applications are tested and approved prior to final deployment to the production environment.

## 3.4 Configuration Files

Although the location and design of configuration schemes will vary according to the particular application server technology, all application servers will include some form of configuration storage mechanism. J2EE servers commonly store their configuration parameters in xml-based configuration files stored on the file system. Applications residing on the application server will also be configurable in the same manner, although some application developers will utilize what is called "annotation" during development, which will embed configuration information directly within their code. This makes the application deployment process more streamlined. Care must be taken to ensure that, along with application files, related configuration files are also protected. When configuration files are utilized, this may be achieved via the use of restrictive file permissions controlled and configured at the OS level.

## 3.5 Users, Groups, Roles, and Realms

Access to the application server must be limited to only those users who have been granted the authority to access the server and the hosted applications. An application server user is similar to an OS user in context; however, the two types of users are separate and distinct. Although the application server resides on top of the OS, the authentication mechanism utilized by the application server will, in most cases, have no knowledge of the user name used to authenticate to the OS. Administrative users who access the application server for management purposes are not to be confused with application users who are granted access to the applications residing on the application server.

The term *Realm* is used within the Java application server to represent a collection of users and groups that are controlled by the same authentication policy. For example, a realm named " certificate authentication" could be created and configured to ensure authentication is certificate based. Users and/or groups associated with that realm would utilize certificate-based authentication methods to authenticate.

A *Group* is a set of authenticated *users*, classified by common traits, defined in the application server.

A *Role* is an abstract name for the permission to access a particular set of resources in an application. A role can be compared to a key that can open a lock. Many people might have a copy of the key. The lock does not care who you are, only that you have the right key. The application designer will identify the roles that will reside within their application, and the application server administrator will map the users and roles to the appropriate security realm.

# 4. GENERAL SECURITY REQUIREMENTS

The capability for secure sessions between application clients and servers is essential. A common method of securing sessions is the use of a protocol that provides data integrity and encryption services. Secure Sockets Layer (SSL) is the most commonly used protocol of this type; most commercial browsers support Versions 2 and 3 of SSL. Transport Layer Security (TLS) is the successor to SSL.

It is a requirement that application servers utilize approved digital certificates and cryptographic modules when protecting application server resources and application data. From an ownership perspective, there are Certificate Authority (CA) certificates, server certificates, application certificates, device certificates, and user certificates. Although the SSL protocol specifies user certificates are optional, security is significantly enhanced by the use of both server and user certificates. The management of digital certificates requires the use of a PKI. A discussion of PKI implementation is beyond the scope of this document. Information can be found in the *Department of Defense Instruction, Department of Defense (DoD) Public Key Infrastructure (PKI) and Public Key (PK) Enabling* documents referenced in *Appendix B: Related Publications*.

Direction on the implementation of certificates on web servers is provided in the *Web Server Security Technical Implementation Guide.*

Direction on secure application development is provided by the *Application Security and Development Security Technical Implementation Guide.*

The presence of specific certificates is especially important with regard to CA certificates. Because these certificates are involved in trust relationships used to confirm identity, it is important to verify that appropriate CA certificates for the DoD PKI are present. Certificates for the DoD PKI External Certificate Authorities (ECAs) and Interim ECAs (IECAs) may be installed at the site's option.

## 4.1 Open Source Software Policy

Many application server products are offered as open source solutions. As such, it must be noted that the DoD has clarified policy on the use of open source software (OSS). This clarification was done in order to take advantage of the capabilities available in the open source community, as long as certain prerequisites are met. The DoD no longer requires that application software be obtained through a valid vendor channel and have a formal support path if the source code for the operating system is publicly available for review.

The following excerpt from DoD memorandum 2009OSS - Clarifying Guidance Regarding Open Source Software (OSS) states:

(2) … Ultimately, the software that best meets the needs and mission of the Department should be used, regardless of whether the software is open source.

*c. DoD Instruction 8500.2, "Information Assurance (IA) Implementation," (reference (g)) includes an Information Assurance Control, "DCPD-1 Public Domain Software Controls," which limits the use of "binary or machine-executable public domain software or other software products with limited or no warranty," on the grounds that these items are difficult or impossible to review, repair, or extend, given that the Government does not have access to the original source code and there is no owner who could make such repairs on behalf of the government. This control should not be interpreted as forbidding the use of OSS, as the source code is available for review, repair and extension by the government and its contractors.*
*d. The use of any software without appropriate maintenance and support presents an information assurance risk. Before approving the use of software (including OSS), system/program managers, and ultimately Designated Approving Authorities (DAAs), must ensure that the plan for software support (e.g., commercial or Government program office support) is adequate for mission need.*

## 4.2 Configuration Management

Configuration management is often overlooked when securing an application server. It is important to have the ability to consistently install a baselined system to production without error or security flaws. Two common methods for building a server from a secure baseline are by following a well-documented build process or by using virtual machine (VM) images.

During the operation of the application server, proper testing of patches and upgrades must be performed. Maintaining a documented process for patch testing and implementation assists in a smooth transition from the lab environment to the production environment.

Configuration management also includes performing a risk assessment of the web server and hosted applications and then implementing a risk management plan. During the assessment, the impact to the organization of an application server failure is analyzed and contingency planning is performed. During the analysis, plans are formed for tasks such as how often backups are performed and where the backups will be stored, and backup equipment and locations are procured. When the application server is a high-priority system, web clustering may be implemented and disaster exercises performed to limit the exposure to a Denial of Service (DoS) during a failure either due to hardware, natural disaster, human error, or a security incident.

## 4.3 Software Installation

Determining the services and software that is needed for the proper operation of both the application server and the hosted application should be completed first, and then the application server software can be installed.

Too often, application servers are installed with the default settings and configurations, demo software, and unnecessary services, plug-ins, and modules. These unneeded services or bits of code can potentially introduce an unintended attack vector which may not be discovered or fixed since these services are neither part of the daily patch routine nor are they used during normal server operation. Removing unneeded services and software post-installation is more involved

and the potential to overlook unnecessary software is greater than properly configuring the application server pre-installation.