

UNCLASSIFIED



# **APACHE TOMCAT APPLICATION SERVER 9 SECURITY TECHNICAL IMPLEMENTATION GUIDE (STIG) OVERVIEW**

**Version 3, Release 4**

**01 April 2026**

**Developed by DISA for the DOD**

UNCLASSIFIED

### **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

	<b>Page</b>
<b>1. INTRODUCTION.....</b>	<b>1</b>
1.1 Executive Summary.....	1
1.2 Authority.....	1
1.3 Vulnerability Severity Category Code Definitions.....	2
1.4 STIG Distribution.....	2
1.5 Document Revisions.....	2
1.6 Other Considerations.....	2
1.7 Product Approval Disclaimer.....	3
<b>2. ASSESSMENT CONSIDERATIONS.....</b>	<b>4</b>
2.1 Security Assessment Information.....	4
2.1.1 Tomcat Environment Variables.....	4
2.1.2 Setenv.sh and Systemd.....	4
2.1.3 OpenJDK and Java Environment Variables.....	4
<b>3. CONCEPTS AND TERMINOLOGY CONVENTIONS.....</b>	<b>6</b>
3.1 Tomcat's Architecture.....	6

## LIST OF TABLES

	<b>Page</b>
Table 1-1: Vulnerability Severity Category Code Definitions .....	2

## LIST OF FIGURES

	<b>Page</b>
Figure 3-1: Tomcat Architecture .....	6

## 1. INTRODUCTION

### 1.1 Executive Summary

The Apache Tomcat Application Server 9 Security Technical Implementation Guide (STIG) is published as a tool to improve the security of Department of Defense (DOD) information systems. This document is meant for use in conjunction with other STIGs such as the Enclave, Network Infrastructure, Secure Remote Computing, and appropriate operating system (OS) STIGs.

Apache Tomcat is a popular open-source application server that provides a platform for Java Servlet, JavaServer Pages, Java Expression Language, and WebSocket technologies. Tomcat also provides a web server environment where Java application code can be deployed and run. The Tomcat web server component is designed to work in conjunction with separate Apache web servers, which are often used when a reverse proxy web application architecture is required.

The Tomcat STIG was developed using Apache Tomcat 9 version 9.0.22 and Ubuntu's OpenJDK 11.0.4+11 running on the Ubuntu Linux 18.04 bionic OS version. When applying the STIG to other Linux flavors, the SME must adapt the STIG file path information and commands to those used by the flavor of Linux being assessed. Bear in mind that the STIG was written for the management of the base Tomcat server. Applications provided with Tomcat that are used for managing the Tomcat server (e.g., the manager app) are within scope, while applications installed on the Tomcat server are not in scope.

Tomcat instances operating on the Windows operating system are also not in scope for this STIG. Due to the availability of Tomcat source code and the open-source nature of the product, Tomcat is often embedded into other products, particularly multitiered client server web applications. Using the Tomcat source code in their product allows vendors to quickly incorporate Java-based web application features and functions without recreating or licensing a separate application server component. Embedded Tomcat configurations contained within application code are not within scope of the STIG development effort. While some of the STIG requirements could potentially be applied in some of these instances, this is a case-by-case scenario. The configuration steps within the STIG were designed and tested against the publicly available Tomcat product that is downloaded and installed as a distinct application server.

### 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.

### 1.3 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 <b>directly and immediately</b> result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which <b>has a potential</b> to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which <b>degrades measures</b> to protect against loss of Confidentiality, Availability, or Integrity.

### 1.4 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/>.

### 1.5 Document Revisions

Comments or proposed revisions to this document should be sent via email to the following address: [disa.stig\\_spt@mail.mil](mailto: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 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.

## 1.7 Product Approval Disclaimer

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](mailto: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 Security Assessment Information

#### 2.1.1 Tomcat Environment Variables

The \$CATALINA\_HOME and \$CATALINA\_BASE environment variables identify the Tomcat home folder locations and which corresponding libraries, configuration settings, and binaries are used at runtime. \$CATALINA\_HOME is a required variable setting, \$CATALINA\_BASE is optional and is employed when multiple instances of Tomcat are in use.

If Tomcat has not been configured for multiple instances by setting a CATALINA\_BASE directory, then \$CATALINA\_BASE will be set to the value of \$CATALINA\_HOME, which is the directory into which Tomcat has been installed. When applying the STIG to a system with multiple Tomcat instances, the reviewer or SME applying the STIG must take this into account and ensure the \$CATALINA\_BASE variable value is in alignment with the installation being assessed.

#### 2.1.2 Setenv.sh and Systemd

System administrators have the option to use a “setenv.sh” script to configure Tomcat environment variables on startup, although this is not provided with the Tomcat distribution package. The setenv.sh script is a shell script that can be created by the system administrator and placed into the \$CATALINA\_HOME/bin folder or the \$CATALINA\_BASE/bin folder, depending on whether the server is running one instance or multiple instances of Tomcat. Setenv.sh can then be used to set or change Tomcat environment variables to suit specific installations. Setenv.sh is only used if shell scripts are used for starting Tomcat.

The Tomcat STIG was developed using Ubuntu 18.04 (Bionic Beaver), which uses systemd rather than init.d for process initialization. Therefore, a setenv.sh script is not used or referenced in the STIG. Reviewers and system administrators must bear this fact in mind when applying the STIG. If the Tomcat server being reviewed does not use systemd to manage the Tomcat processes, the reviewer or system administrator will need to modify STIG check and fix language to accommodate a non-systemd based system.

#### 2.1.3 OpenJDK and Java Environment Variables

Tomcat 9 requires a Java Runtime Environment conforming to JRE 8 or later. Due to architectural changes with Java version 11, a separate JRE is no longer available; therefore, OpenJDK version 11 was used for STIG testing. Tools are available that allow administrators to create a distinct JRE distribution based on the JDK; however, the Tomcat STIG does not implement or require a specific DOD enterprise-wide JRE that must be installed on all systems. Therefore, the version of Java used on the system being assessed can potentially be different from the OpenJDK version that was used during the STIG development effort. The STIG cannot factor in every possibility, which means some check and fix commands may have to be modified during the application of the STIG to account for differences with the specific Java version being used.

In addition to Tomcat’s environment variables, Java environment variables that specify the Java home folders and Java binaries must also be configured on startup. The STIG was written using the

`$JAVA_HOME` variable rather than the `$JRE_HOME` variable since the Java Development Kit was used in testing. These variables could change on a system-by-system basis depending on what Java runtime is used on the system to which the STIG is being applied. The SME or reviewer applying the STIG must account for these differences in their specific deployment and modify the STIG check and fix language accordingly.

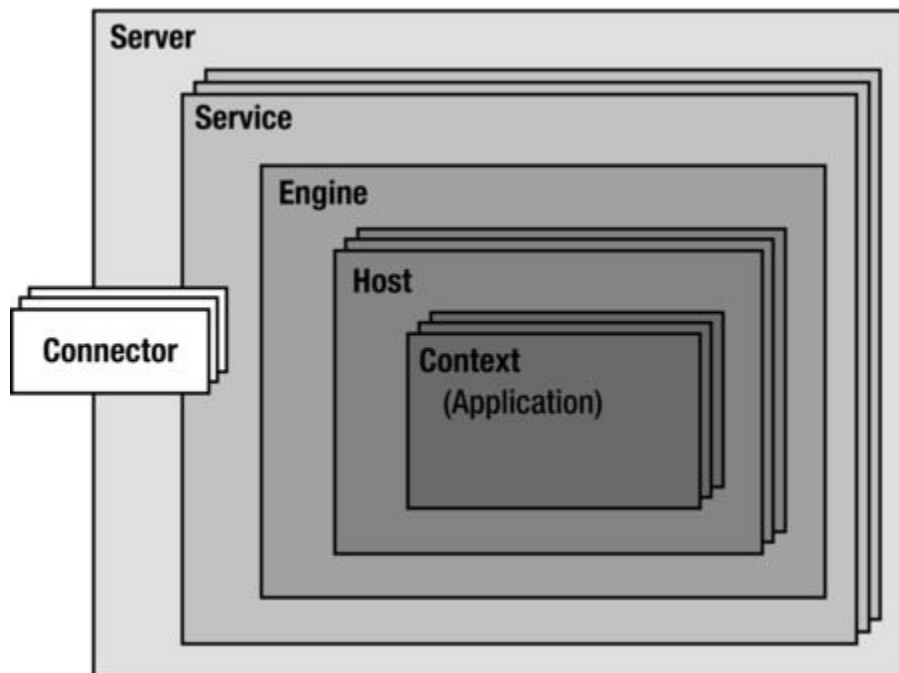
The STIG was written and tested on Ubuntu 18.04 and uses `systemd` for the Tomcat startup scripts. While the `systemd` startup scripts for Tomcat could technically be named anything, the STIG uses “tomcat.service” as the name of the Tomcat startup script. The startup script contains the Java environment variables that must be configured as per the STIG.

### 3. CONCEPTS AND TERMINOLOGY CONVENTIONS

#### 3.1 Tomcat's Architecture

**Tomcat's architecture** consists of a series of functional components that can be combined according to well-defined rules.

Figure 3-1: Tomcat Architecture



The structure of each server installation (via these functional components) is defined in the file `server.xml`, which is located in the `/conf` subdirectory of **Tomcat's installation folder**. This file contains many of the STIG configuration parameters.

Tomcat configuration files are formatted as schemaless XML; elements and attributes are case-sensitive. The configuration element descriptions are organized into the following major categories:

- **Top-Level Elements** – `<Server>` is the root element of the entire configuration file, while `<Service>` represents a group of Connectors that is associated with an Engine.
- **Connectors** – Represent the interface between external clients sending requests to (and receiving responses from) a particular Service.
- **Containers** – Represent components whose function is to process incoming requests and create the corresponding responses. An Engine handles all requests for a Service, a Host handles all requests for a particular virtual host, and a Context handles all requests for a specific web application.
- **Nested Components** – Represent elements that can be nested inside the element for a Container. Some elements can be nested inside any Container, while others can only be nested inside a Context.

The **Context** element represents a *web application*, which is run within a particular virtual host. Each web application is based on a *Web Application Archive* (WAR) file, or a corresponding directory containing the corresponding unpacked contents, as described in the Servlet Specification (version 2.2 or later). For more information about web application archives, download the [Servlet Specification](#) and review the Tomcat [Application Developer's Guide](#).

The web application used to process each HTTP request is selected by Catalina based on matching the longest possible prefix of the Request URI against the *context path* of each defined Context. Once selected, that Context will select an appropriate servlet to process the incoming request according to the servlet mappings defined by the web application deployment.

You may define as many **Context** elements as you wish. Each such Context **MUST** have a unique context name within a virtual host. The context path does not need to be unique (see *parallel deployment* below). In addition, a Context must be present with a context path equal to a zero-length string. This Context becomes the *default* web application for this virtual host and is used to process all requests that do not match any other Context's context path.

The **Engine** element represents the entire request processing machinery associated with a particular Catalina [Service](#). It receives and processes *all* requests from one or more **Connectors** and returns the completed response to the Connector for ultimate transmission back to the client. Exactly one **Engine** element **MUST** be nested inside a [Service](#) element, following all of the corresponding Connector elements associated with this Service.

The **Host** element represents a *virtual host*, which is an association of a network name for a server (such as "www.mycompany.com") with the particular server on which Tomcat is running. For clients to be able to connect to a Tomcat server using its network name, this name must be registered in the Domain Name Service (DNS) server that manages the Internet domain you belong to.

One or more **Host** elements are nested inside an [Engine](#) element. Inside the Host element, you can nest [Context](#) elements for the web applications associated with this virtual host. Exactly one of the Hosts associated with each Engine **MUST** have a name matching the defaultHost attribute of that Engine.