Fortify Software Security Content

2023 Update 3
September 29, 2023

About OpenText Fortify Software Security Research

The Fortify Software Security Research team translates cutting-edge research into security intelligence that powers the Fortify product portfolio – including Fortify Static Code Analyzer (SCA) and Fortify WebInspect. Today, Fortify Software Security Content supports 1,627 vulnerability categories across 33+ languages and spans more than one million individual APIs.

Fortify Software Security Research (SSR) is pleased to announce the immediate availability of updates to Fortify Secure Coding Rulepacks (English language, version 2023.3.0), Fortify WebInspect SecureBase (available via SmartUpdate), and Fortify Premium Content.

Fortify Secure Coding Rulepacks [Fortify Static Code Analyzer]

With this release, the Fortify Secure Coding Rulepacks detect 1,403 unique categories of vulnerabilities across 33+ languages and span over one million individual APIs. In summary, this release includes the following:

Improved Support for Android 13 (version supported: 33)

The Android platform is an open-source software stack designed for mobile devices. A primary component of Android is the Java API Framework, which exposes Android features to application developers. This release expands vulnerability detection in native Android applications written in Java or Kotlin that leverage Android's Java API Framework. The following three new weakness categories are introduced in this release for Android applications:

- Intent Manipulation: Implicit Internal Intent
- Intent Manipulation: Implicit Pending Intent
- Intent Manipulation: Mutable Pending Intent

Initial Support for Android Jetpack (AndroidX)

Android Jetpack is a set of libraries, tools, and guidance that helps developers create Android applications with greater ease. Jetpack covers the androidx.* packages and is unbundled from platform APIs, which helps facilitate backwards compatibility and allows for more frequent updates. In this release, we provide initial coverage for this software suite.

Initial coverage for Android Jetpack supports detection of weaknesses in the following libraries:

- androidx.appsearch (version supported: 1.1.0-alpha03)
- androidx.compose.foundation (version supported: 1.5.1)
- androidx.compose.material (version supported: 1.5.1)
- androidx.compose.material3 (version supported: 1.1.2)
- androidx.compose.ui (version supported: 1.5.1)
- androidx.core (version supported: 1.12.0)
- androidx.credentials (version supported: 1.2.0-beta04)
- androidx.datastore (version supported: 1.0.0)
- androidx.security.crypto (version supported: 1.0.0)
- androidx.sqlite (version supported: 2.3.1)

Example category coverage improvements include the following:

- Access Control: Database
- Command Injection
- Denial of Service
- Denial of Service: Regular Expression
- Header Manipulation

- Insecure Transport
- Open Redirect
- Password Management: Empty Password
- Password Management: Hardcoded Password
- Path Manipulation
- Privacy Violation
- Resource Injection
- Server-Side Request Forgery
- SQL Injection
- System Information Leak
- System Information Leak: Internal

MySQL Connector/Python Support (version supported: 8.1.0)

MySQL Connector/Python is a software library that facilitates the interaction between Python applications and MySQL databases. It serves as a bridge or connector between the Python programming language and the MySQL database management system, enabling developers to easily connect, query, and manipulate data in MySQL databases using Python code.

Improved category coverage includes the following:

- Access Control: Database
- Denial of Service
- Insecure Transport: Client Identity Verification Disabled
- Insecure Transport: Database
- Insecure Transport: Weak SSL Protocol
- Password Management
- Password Management: Empty Password
- Password Management: Hardcoded Password
- Password Management: Weak Cryptography
- Path Manipulation
- Server-Side Request Forgery
- SQL Injection

Improved Support for Django (version supported: 3.2)

Django is a web framework written in Python that is designed to facilitate secure and rapid web development. Speed and security of development are attained by the high level of abstraction in the framework, where code constructs and generation are used to drastically cut back on boilerplate code. In this release, we update our existing Django coverage to support releases up to version 3.2.

Improved coverage includes the following namespaces: *Django.contrib.auth.models, Django.db.models,* and *Django.http.response*. Additionally, improved coverage of weakness categories includes the following:

- Cookie Security: Overly Permissive SameSite Attribute
- Header Manipulation
- Password Management
- Password Management: Empty Password

- Password Management: Hardcoded Password
- Password Management: Null Password
- Password Management: Weak Cryptography
- Privacy Violation
- System Information Leak
- System Information Leak: External

Initial Support for Bicep (version supported: 0.21.1)[1]

Microsoft Bicep is an open-source domain-specific language (DSL) for Infrastructure-as-Code (IaC) solutions developed by Microsoft to simplify and streamline the deployment of Azure resources. It serves as an abstraction layer on top of Azure Resource Manager (ARM) templates, offering a more intuitive and readable way to define and manage Azure infrastructure. With Bicep, users can write concise and human-readable code to describe Azure resources, configurations, and dependencies.

Initial coverage of weakness categories includes the following:

- Azure ARM Misconfiguration: Automation Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Batch Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Cognitive Services Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Databricks Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Event Hubs Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Hardcoded Secret
- Azure ARM Misconfiguration: HTTPS Not Required
- Azure ARM Misconfiguration: Improper AKS Network Access Control
- Azure ARM Misconfiguration: Improper App Service Access Control
- Azure ARM Misconfiguration: Improper Blob Storage Access Control
- Azure ARM Misconfiguration: Improper Compute VM Access Control
- Azure ARM Misconfiguration: Improper Container Registry Network Access Control
- Azure ARM Misconfiguration: Improper CORS Policy
- Azure ARM Misconfiguration: Improper Custom Role Access Control Policy
- Azure ARM Misconfiguration: Improper DocumentDB Network Access Control
- Azure ARM Misconfiguration: Improper KeyVault Access Control Policy
- Azure ARM Misconfiguration: Improper Security Group Network Access Control
- Azure ARM Misconfiguration: Improper SQL Server Network Access Control
- Azure ARM Misconfiguration: Improper Storage Network Access Control
- Azure ARM Misconfiguration: Insecure Active Directory Domain Service Transport
- Azure ARM Misconfiguration: Insecure App Service Transport
- Azure ARM Misconfiguration: Insecure CDN Transport
- Azure ARM Misconfiguration: Insecure Database for MySQL Storage
- Azure ARM Misconfiguration: Insecure Database for PostgreSQL Storage
- Azure ARM Misconfiguration: Insecure DataBricks Storage
- Azure ARM Misconfiguration: Insecure EventHub Storage
- Azure ARM Misconfiguration: Insecure EventHub Transport
- Azure ARM Misconfiguration: Insecure IoT Hub Transport
- Azure ARM Misconfiguration: Insecure MySQL Server Transport
- Azure ARM Misconfiguration: Insecure PostgreSQL Server Transport

[1] Requires Fortify Static Code Analyzer 23.2.0 and later. Initial security content for Bicep is distributed with Fortify Static Code Analyzer 23.2.x.

- Azure ARM Misconfiguration: Insecure Recovery Services Backup Storage
- Azure ARM Misconfiguration: Insecure Recovery Services Vaults Storage
- Azure ARM Misconfiguration: Insecure Redis Enterprise Transport
- Azure ARM Misconfiguration: Insecure Redis Transport
- Azure ARM Misconfiguration: Insecure Service Bus Storage
- Azure ARM Misconfiguration: Insecure Service Bus Transport
- Azure ARM Misconfiguration: Insecure Storage Account Storage
- Azure ARM Misconfiguration: Insecure Storage Account Transport
- Azure ARM Misconfiguration: Insufficient AKS Monitoring
- Azure ARM Misconfiguration: Insufficient Application Insights Logging
- Azure ARM Misconfiguration: Insufficient Application Insights Monitoring
- Azure ARM Misconfiguration: Insufficient Microsoft Defender Monitoring
- Azure ARM Misconfiguration: Insufficient SQL Server Logging
- Azure ARM Misconfiguration: Insufficient SQL Server Monitoring
- Azure ARM Misconfiguration: IoT Hub Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: NetApp Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Public Access Allowed
- Azure ARM Misconfiguration: Service Bus Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Storage Account Missing Customer-Managed Encryption Key
- Azure ARM Misconfiguration: Weak App Service Authentication
- Azure ARM Misconfiguration: Weak SignalR Authentication
- Password Management: Empty Password
- Password Management: Hardcoded Password
- Privacy Violation
- Privacy Violation: Missing Secure Decorator

Initial Support for Solidity (version supported: 0.8.x)[2]

Solidity is an object-oriented programming language used for developing smart contracts in various decentralized blockchain environments, most notably, in the Ethereum blockchain. Smart contracts written in Solidity run mainly on an Ethereum Virtual Machine (EVM) but can also run on other compatible virtual machines.

Initial coverage of weakness categories includes the following:

- Authorization Bypass: tx.origin
- Code Correctness: Failing Assertion
- Code Correctness: Reentrancy
- Code Correctness: Typographical Error
- Dead Code
- Denial of Service: External Call
- Dynamic Code Evaluation: Delegatecall
- Integer Overflow
- Obsolete
- Often Misused: Block Values
- Poor Style: Confusing Naming

[2] Requires Fortify Static Code Analyzer 23.2.0 and later. Initial security content for Solidty is distributed with Fortify Static Code Analyzer 23.2.x.

- Poor Style: Variable Never Used
- Solidity Bad Practices: Default Function Visibility
- Solidity Bad Practices: Ether Balance Check
- Solidity Bad Practices: Hardcoded Gas Amount
- Solidity Bad Practices: Lack of Explicit Variable Visibility
- Solidity Bad Practices: Missing Constructor
- Solidity Misconfiguration: Compiler With Known Vulnerabilities
- Solidity Misconfiguration: Floating Pragma
- Unchecked Return Value
- Uninitialized Variable

Cloud Infrastructure as Code (IaC)

Infrastructure as code is the process of managing and provisioning computer resources through code, rather than various manual processes. Expanded coverage of supported technologies include Terraform configurations for deployment to Microsoft Azure, as well as configurations for AWS Ansible. Common issues related to the configuration of these services mentioned are now reported to the developer.

Microsoft Azure Terraform Configurations

Terraform is an open-source IaC tool for building, changing and versioning cloud infrastructure. It uses its own declarative language known as HashiCorp Configuration Language (HCL). Cloud infrastructure is codified in configuration files to describe the desired state. Terraform providers support the configuration and management of Microsoft Azure infrastructure. Improved coverage of weakness categories includes the following for Terraform configurations:

- Azure Terraform Misconfiguration: App Service Auto Upgrade Disabled
- Azure Terraform Misconfiguration: Improper AKS Access Control
- Azure Terraform Misconfiguration: Improper AKS Network Access Control
- Azure Terraform Misconfiguration: Improper App Service Access Control
- Azure Terraform Misconfiguration: Improper Cognitive Search Network Access Control
- Azure Terraform Misconfiguration: Improper Container Registry Access Control
- Azure Terraform Misconfiguration: Improper Functions Access Control
- Azure Terraform Misconfiguration: Improper MariaDB Network Access Control
- Azure Terraform Misconfiguration: Improper MySQL Network Access Control
- Azure Terraform Misconfiguration: Improper SQL Database Network Access Control
- Azure Terraform Misconfiguration: Improper Storage Account Access Control
- Azure Terraform Misconfiguration: Improper Virtual Network Access Control
- Azure Terraform Misconfiguration: Insecure Disk Storage
- Azure Terraform Misconfiguration: Insecure PostgreSQL Storage
- Azure Terraform Misconfiguration: Insufficient AKS Monitoring
- Azure Terraform Misconfiguration: Insufficient Application Gateway Monitoring
- Azure Terraform Misconfiguration: Insufficient Defender for Cloud Monitoring
- Azure Terraform Misconfiguration: Insufficient Front Door Monitoring
- Azure Terraform Misconfiguration: Insufficient MariaDB Backup
- Azure Terraform Misconfiguration: Insufficient Monitor Logging
- Azure Terraform Misconfiguration: Insufficient Network Watcher Logging
- Azure Terraform Misconfiguration: Insufficient PostgreSQL Monitoring
- Azure Terraform Misconfiguration: Insufficient SQL Database Monitoring
- Azure Terraform Misconfiguration: Redis Cache Auto Upgrade Disabled
- Azure Terraform Misconfiguration: Reduced Virtual Network Availability

- Azure Terraform Misconfiguration: Weak App Service Authentication
- Azure Terraform Misconfiguration: Weak Functions Authentication
- Azure Terraform Misconfiguration: Weak Linux Virtual Machines Authentication
- Azure Terraform Misconfiguration: Weak Service Fabric Authentication

Amazon Web Services (AWS) Ansible Configurations

Ansible is an open-source automation tool that provides configuration management, application deployment, cloud provisioning, and node orchestration to various environments. Ansible includes modules that support the configuration and management of Amazon Web Services (AWS). Improved coverage of weakness categories includes the following for AWS Ansible configurations:

- AWS Ansible Misconfiguration: EFS Missing Customer-Managed Encryption Key
- AWS Ansible Misconfiguration: Improper API Gateway Network Access Control
- AWS Ansible Misconfiguration: Improper ECR Access Control
- AWS Ansible Misconfiguration: Improper ECS Network Access Control
- AWS Ansible Misconfiguration: Improper S3 Access Control
- AWS Ansible Misconfiguration: Improper Stack Access Control
- AWS Ansible Misconfiguration: Insecure API Gateway Transport
- AWS Ansible Misconfiguration: Insecure CloudFront Transport
- AWS Ansible Misconfiguration: Insecure CloudTrail Storage
- AWS Ansible Misconfiguration: Insecure CodeBuild Storage
- AWS Ansible Misconfiguration: Insecure RDS Transport
- AWS Ansible Misconfiguration: Insufficient API Gateway Logging
- AWS Ansible Misconfiguration: Insufficient CloudFront Logging
- AWS Ansible Misconfiguration: Insufficient Lambda Logging
- AWS Ansible Misconfiguration: Insufficient RDS Backup
- AWS Ansible Misconfiguration: Insufficient S3 Backup
- AWS Ansible Misconfiguration: Insufficient S3 Logging
- AWS Ansible Misconfiguration: Insufficient S3 Monitoring
- AWS Ansible Misconfiguration: Insufficient Stack Monitoring
- AWS Ansible Misconfiguration: Privileged Batch Container
- AWS Ansible Misconfiguration: RDS Auto-Upgrade Disabled
- AWS Ansible Misconfiguration: RDS Missing Customer-Managed Encryption Key
- AWS Ansible Misconfiguration: Reduced CloudFront Availability
- AWS Ansible Misconfiguration: Reduced EC2 Availability
- AWS Ansible Misconfiguration: Reduced ELB Availability
- AWS Ansible Misconfiguration: Weak IAM Password Policy

2023 Common Weakness Enumeration (CWE™) Top 25

The Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Weaknesses (CWE Top 25) was introduced in 2019 and replaces SANS Top 25. Released in June of 2023, the 2023 CWE Top 25 is determined using a heuristic formula that normalizes the frequency and severity of vulnerabilities reported to the National Vulnerability Database (NVD) over the past two years. To support our customers who want to prioritize their auditing around the most commonly reported critical vulnerabilities in the NVD, a correlation of the Fortify Taxonomy to the 2023 CWE Top 25 has been added.

OWASP API Security Top 10 2023

The Open Worldwide Application Security Project (OWASP) API Security Top 10 2023 provides a list of the top security risks affecting APIs in 2023. It aims to raise awareness around API security weaknesses and to educate those involved in API development and maintenance, such as developers, designers, architects, managers and/or organizations in general who need to secure Web APIs.

The OWASP API Security Top 10 focuses on weaknesses affecting Web APIs and it is not intended to be used only by itself, instead it is intended to be used in combination with other standards and best practices in order to thoroughly capture all relevant risks. For example: it should be used in combination with the OWASP Top 10 in order to identify issues related to input validation such as injections. To support our customers who want to mitigate Web Application risk, correlation of the Fortify Taxonomy to the newly released OWASP API Security Top 10 2023 has been added.

Center for Internet Security (CIS) Benchmarks

The Center for Internet Security (CIS) benchmarks are a collection of community-developed secure configuration recommendations that are mapped to the CIS Critical Security Controls. These recommendations are intended to enable securing cloud infrastructure as well as demonstrate compliance with industry standards. The CIS benchmarks are continuously updated in order to adapt to evolving state of cybersecurity for the 25+ vendor product families covered. Product families supported include the following:

- Amazon Elastic Kubernetes Service (EKS) Benchmark v1.3.0
- Amazon Web Services Foundations Benchmark v2.0.0
- Azure Kubernetes Service (AKS) Benchmark v1.3.0
- Google Cloud Computing Platform Benchmark v2.0.0
- Google Kubernetes Engine (GKE) Benchmark v1.4.0
- Kubernetes Benchmark v1.7.1
- Microsoft Azure Foundations Benchmark v2.0.0

Smart Contract Weakness Classification (SWC)[3]

Smart Contract Weakness Classification (SWC) is a systematic framework that categorizes and explains vulnerabilities in smart contracts. It provides a standardized way to understand and address weaknesses in these self-executing code pieces running on blockchains like Ethereum. Notably, the SWC registry's content has not been comprehensively updated since 2020, resulting in known incompleteness, errors, and important omissions. To support our customers who want to mitigate risks in smart contracts, correlation of the Fortify Taxonomy to the current version of SWC has been added.

[3] Requires scan from Fortify Static Code Analyzer 23.2.0 and later.

Miscellaneous Errata

In this release, resources have been invested to ensure we can reduce the number of false positive issues, refactor for consistency, and improve the ability for customers to audit issues. Customers can also expect to see changes in reported issues related to the following:

Deprecation of Fortify Static Code Analyzer Versions Prior to 20.x

As observed with the 2022.4 release, we are continuing to support the last four major releases of Fortify Static Code Analyzer. Therefore, this will be the last release of the Rulepacks that support Fortify Static Code Analyzer versions prior to 20.x. For the next release, Fortify Static Code Analyzer versions prior to 20.x will not load the most recent Rulepacks. This will require either downgrading the Rulepacks or upgrading the version of Fortify Static Code Analyzer. For future releases, we will continue to support the last four major releases of Fortify Static Code Analyzer.

False Positive Improvements

Work has continued with the effort to remove false positives in this release. In addition to other improvements, customers can expect further removal of false positives in the following areas:

- *ASP.NET MVC Bad Practices: Optional Submodel With Required Property* – false positives removed related to virtual fields in ASP.NET applications
- *Code Correctness: Double-Checked Locking* – false positives removed in Java applications
- *Cross-Site Request Forgery* – false positives removed for HTML forms using `AntiForgery.GetHtml()` or `Html.AntiForgeryToken()` in .NET applications
- *Cross-Site Scripting: Persistent* – false positives removed related to the `cycle` tag in Django applications
- *Double Free* – false positives removed in C/C++ applications that use `throw_error()` from the boost library
- *HTML5: Missing Content Security Policy* – false positives removed in Java applications
- *JSON Injection* – false positives removed in PHP applications
- *Mass Assignment: Insecure Binder Configuration* – false positives removed related to Enum types in .NET applications
- *Often Misused: File System* – false positives removed related to `GetFullPathNameW()` and similar function calls in C++ applications
- *Path Manipulation* – false positives removed in Java applications using the Amazon AWS SDK
- *Type Mismatch: Signed to Unsigned* – false positives removed relating to boolean values in C/C++ applications
- *Unreleased Resource* – false positives removed when using `CreateFileW()` in C++ applications

Category Changes

When weakness category name changes occur, analysis results when merging prior scans with new scans will result in added/removed categories.

To improve consistency, the following 14 categories have been renamed:

Removed Category	Added Category
AWS CloudFormation Misconfiguration: Insecure Elasticache Storage	AWS CloudFormation Misconfiguration: Insecure ElastiCache Storage
AWS CloudFormation Misconfiguration: Insecure Elasticache Transport	AWS CloudFormation Misconfiguration: Insecure ElastiCache Transport
AWS Terraform Misconfiguration: Elasticache Missing Customer-Managed Encryption Key	AWS Terraform Misconfiguration: ElastiCache Missing Customer-Managed Encryption Key
Azure Terraform Bad Practices: AKS Cluster Missing Host-Based Encryption	Azure Terraform Misconfiguration: AKS Cluster Missing Host-Based Encryption
Azure Terraform Bad Practices: Azure MySQL Server Missing Infrastructure Encryption	Azure Terraform Misconfiguration: MySQL Missing Infrastructure Encryption
Azure Terraform Bad Practices: Azure PostgreSQL Server Missing Infrastructure Encryption	Azure Terraform Misconfiguration: PostgreSQL Missing Infrastructure Encryption
Azure Terraform Bad Practices: Missing Azure Storage Infrastructure Encryption	Azure Terraform Misconfiguration: Storage Account Missing Infrastructure Encryption
Azure Terraform Bad Practices: Missing SQL Database Backup Encryption	Azure Terraform Misconfiguration: SQL Server Backup Missing Encryption
Azure Terraform Bad Practices: Scale Set Missing Host-Based Encryption	Azure Terraform Misconfiguration: Scale Set Missing Host-Based Encryption
Azure Terraform Bad Practices: VM Missing Host-Based Encryption	Azure Terraform Misconfiguration: VM Missing Host-Based Encryption
GCP Terraform Bad Practices: Overly Permissive Service Account	GCP Terraform Misconfiguration: Improper Compute Engine Access Control
GCP Terraform Misconfiguration: Weak Key Management	GCP Terraform Misconfiguration: Compute Engine Missing Customer-Managed Encryption Key
Kubernetes Bad Practices: Improper Admission Controller Access Control	Kubernetes Misconfiguration: Improper Admission Controller Access Control
Kubernetes Misconfiguration: Missing Service Account Admission Controller	Kubernetes Misconfiguration: Missing ServiceAccount Admission Controller

Fortify Priority Order Changes

To improve consistency across vulnerability categories related to missing customer-managed encryption keys, the Fortify Priority Order of the following 20 categories has been changed to "low":

- *Azure ARM Misconfiguration: Automation Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: Batch Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: Cognitive Services Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: Databricks Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: Event Hubs Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: IoT Hub Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: NetApp Missing Customer-Managed Encryption Key*
- *Azure ARM Misconfiguration: Service Bus Missing Customer-Managed Encryption Key*

- *Azure ARM Misconfiguration: Storage Account Missing Customer-Managed Encryption Key*
- *Azure Terraform Misconfiguration: AKS Cluster Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Azure Disk Snapshot Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Container Registry Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Cosmos DB Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Managed Disk Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Shared Image Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: SQL Database Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Storage Account Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: Storage Encryption Scope Missing Customer-Managed Key*
- *Azure Terraform Misconfiguration: VM Storage Missing Customer-Managed Key*
- *GCP Terraform Misconfiguration: Compute Engine Missing Customer-Managed Encryption Key*

Fortify SecureBase [Fortify WebInspect]

Fortify SecureBase combines checks for thousands of vulnerabilities with policies that guide users in the following updates available immediately via SmartUpdate.

Vulnerability Support

Insecure Deployment: Unpatched Application

A pre-authorization Remote Code Execution (RCE) vulnerability in vBulletin versions 5.6.0 through 5.6.8 has been identified by CVE-2023-25135. vBulletin, a popular software for building dynamic online communities and forums, improperly sanitizes user-provided input for unauthenticated deserialization. This issue enables attackers to execute arbitrary code on the server, abuse application logic, or mount Denial of Service (DoS) attacks. This release includes a check to detect this vulnerability on target servers.

Prototype Pollution: Server-Side

Server-side prototype pollution occurs when an attacker can manipulate the prototype of an object. This is possible in prototype-based languages such as JavaScript, which enables altering of properties and methods at runtime. Severity of the exploit depends on where the polluted object is used in the application. Attacks include Denial of Service, changing application configuration, and in some cases Remote Code Execution. This release includes a check to detect prototype pollution in web applications.

Compliance Reports

2023 Common Weakness Enumeration (CWE™) Top 25

The Common Weakness Enumeration (CWE™) Top 25 Most Dangerous Software Weaknesses (CWE Top 25) was introduced in 2019 and replaces SANS Top 25. Released in June, the 2023

CWE Top 25 is determined using a heuristic formula that normalizes the frequency and severity of vulnerabilities reported to the National Vulnerability Database (NVD) over the past two years. This SecureBase update includes checks that map either directly to the category identified by the CWE Top 25, or a CWE-ID related to a CWE-ID in the Top 25 via “ChildOf” relationship.

OWASP API Security Top 10 2023

The Open Worldwide Application Security Project (OWASP) API Security Top 10 2023 provides a list of the top security risks affecting APIs in 2023. It aims to raise awareness around API security weaknesses and to educate those involved in API development and maintenance, such as developers, designers, architects, managers, and organizations in general who need to secure Web APIs. The OWASP API Security Top 10 focuses on weaknesses affecting Web APIs and it is not intended to be used by itself. Instead, it’s intended to be used in combination with other standards and best practices to thoroughly capture all relevant risks. For example: Use the OWASP API Security Top 10 2023 in combination with the OWASP Top 10 to identify issues related to input validation such as injections. This SecureBase update includes a new compliance report template that provides correlation between OWASP API Security Top 10 2023 categories and WebInspect checks.

Policy Updates

2023 CWE Top 25

A policy customized to include checks relevant to 2023 CWE Top 25 has been added to the WebInspect SecureBase list of supported policies.

OWASP API Security Top 10 2023

A policy customized to include checks relevant to OWASP API Security Top 10 2023 has been added to the WebInspect SecureBase list of supported policies. This policy contains a subset of the available WebInspect checks that enables customers to run compliance-specific WebInspect scans.

Miscellaneous Errata

In this release, we invested resources to further reduce the number of false positives and improve the ability for customers to audit issues. Customers can also expect to see changes in reported findings related to the following areas.

LDAP Injection

This release includes improvements for the LDAP Injection check to reduce false positives and improve the accuracy of the results.

SSL Certificate Hostname Discrepancy

The SSL Certificate Hostname Discrepancy check report content now includes more detailed information that should help customers apply a proper fix for this security issue.

Aggressive Coverage by Check Inputs

For some WebInspect checks, it is possible to enable Aggressive Coverage that guides WebInspect to send a longer list of attacks that target a wider range of endpoints. This release includes improvements to these checks, which enable customers to configure Aggressive Coverage by changing Check Inputs instead of adding separate checks to the scan policy. The checks that have Aggressive Coverage capabilities include the following: *Log4Shell*, *JNDI Reference Injection*, *Server-Side Request Forgery*, *OS Command Injection*, and *Server-Side Prototype Pollution*. Checks with Aggressive Coverage enabled provide more accurate scanning, however, it is important to consider that the number of requests and the scan time might drastically increase. Therefore, Fortify strongly recommends that you run checks with Aggressive Coverage enabled in a separate policy without other checks.

Web Server Misconfiguration: Unprotected File
This release includes a minor bug fix to improve the detection of Java-related configuration files.

Fortify Premium Content

The research team builds, extends, and maintains a variety of resources outside our core security intelligence products.

2023 CWE Top 25

To accompany the new correlations, this release also contains a new report bundle for Fortify Software Security Center with support for the 2023 CWE Top 25, which is available for download from the Fortify Customer Support Portal under Premium Content.

OWASP API Security Top 10 2023

To accompany the new correlations, this release also contains a new report bundle for Fortify Software Security Center with support for the OWASP API Security Top 10, which is available for download from the Fortify Customer Support Portal under Premium Content.

Fortify Taxonomy: Software Security Errors

The Fortify Taxonomy site, which contains descriptions for newly added category support, is available at https://vulncat.fortify.com.

Contact Fortify Technical Support

OpenText Fortify
https://softwaresupport.softwaregrp.com/
+1 (800) 509-1800

Contact SSR

Alexander M. Hoole
Senior Manager, Software Security Research
OpenText Fortify
hoole@opentext.com
+1 (650) 427-9973

Peter Blay
Manager, Software Security Research
OpenText Fortify
pblay@opentext.com
+1 (669) 309-1634