Author

CAST



**OWASP 2021 TOP 10**

**Summary Report**

Application Name –

Version –

CAST AIP -

|  |
| --- |
|  |
|  |

Monday, xx July 2012

My Application Name

Version Number

My CAST Version

# Table of Content

Table of Content

1. Introduction

1.1. Application Characteristics

2. Security Violation Overview

2.1. OWASP -2021 Top 10 Risk Factor Summary

2.2. OWASP A01:2021 – Broken Access Control

2.3. OWASP A02:2021 – Cryptographic Failures

2.4. OWASP A03:2021 – Injection

2.5. OWASP A04:2021 – Insecure Design

2.6. OWASP A05:2021 – Security Misconfiguration

2.7. OWASP A06:2021 – Vulnerable and Outdated Components

2.8. OWASP A07:2021 – Identification and Authentication Failures

2.9. OWASP A08:2021 – Software and Data Integrity Failures

2.10. OWASP A09:2021 – Security Logging and Monitoring Failures

2.11. OWASP A10:2021 – Server-Side Request Forgery (SSRF)

3. Appendix

3.1. About CAST Software Intelligence

3.2. About CAST Security

# Introduction

This assessment is an effort to determine the security health of the application and identify some of the root causes of current Security concerns, as well as any risks of future degradation. This assessment uses the CAST Application Intelligence Platform (AIP) to automatically scan the implementation of these applications to review the architecture, design, and code against OWASP standards.

CAST AIP adapts the quality rules from best-in-class industry standards (ISO, CWE, OWASP, PCI DSS, STIG, NIST,…). With its unique ability to perform dataflow and system-level analysis (from Presentation layer or REST exposed services down to Database layer), CAST provides the most accurate security findings, reducing a lot of false positives.

## Application Characteristics

This assessment is focused solely on the technical implementation of the said application (user interface to database), with no investigation of the functionality.

|  |  |
| --- | --- |
| Name | Value |
| kLoC | 504 |
| Files | 6,586 |
| Classes | 593 |
| SQL Art. | 0 |
| Tables | 119 |

*Fig 1: Application Technology characteristics Table 1: Application characteristics*

# Security Violation Overview

This section provides a summary of the most severe security vulnerability identified in the structural quality analysis and measurement by CAST AIP against the OWASP Top 10 2021 standard. Details about OWASP Security Standard can be found [here](https://www.owasp.org/index.php/Top_10-2017_Top_10).

## OWASP -2021 Top 10 Risk Factor Summary

The [OWASP Top 10](https://owasp.org/Top10/)focuses on identifying the most serious web application security risks for a broad array of organizations.

The following table presents a summary of the 2021 Top 10 Application Security Risks.



The results of CAST AIP analysis are presented hereafter.

List of OWASP -2021 Top 10 rules that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| OWASP-2021 Top Ten | Total | Added | Removed |
| A01:2021 – Broken Access Control | 0 | 0 | 0 |
| A02:2021 – Cryptographic Failures | 0 | 0 | 0 |
| A03:2021 – Injection | 0 | 0 | 0 |
| A04-2021 - | 0 | 0 | 0 |
| A… | 0 | 0 | 0 |

*Table 2: OWASP 2021 Top Ten Rules*

## OWASP A01:2021 – Broken Access Control

Access control enforces policy such that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure, modification, or destruction of all data or performing a business function outside the user's limits. Common access control vulnerabilities include:

* Violation of the principle of least privilege or deny by default, where access should only be granted for particular capabilities, roles, or users, but is available to anyone.
* Bypassing access control checks by modifying the URL (parameter tampering or force browsing), internal application state, or the HTML page, or by using an attack tool modifying API requests.
* Permitting viewing or editing someone else's account, by providing its unique identifier (insecure direct object references)
* Accessing API with missing access controls for POST, PUT and DELETE.
* Elevation of privilege. Acting as a user without being logged in or acting as an admin when logged in as a user.
* Metadata manipulation, such as replaying or tampering with a JSON Web Token (JWT) access control token, or a cookie or hidden field manipulated to elevate privileges or abusing JWT invalidation.
* CORS misconfiguration allows API access from unauthorized/untrusted origins.
* Force browsing to authenticated pages as an unauthenticated user or to privileged pages as a standard user.

List of A01:2021 – Broken Access Control vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 3:* *A01:2021 – Broken Access Control Vulnerabilities*

## OWASP A02:2021 – Cryptographic Failures

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information, and business secrets require extra protection, mainly if that data falls under privacy laws, e.g., EU's General Data Protection Regulation (GDPR), or regulations, e.g., financial data protection such as PCI Data Security Standard (PCI DSS). For all such data:

* Is any data transmitted in clear text? This concerns protocols such as HTTP, SMTP, FTP also using TLS upgrades like STARTTLS. External internet traffic is hazardous. Verify all internal traffic, e.g., between load balancers, web servers, or back-end systems.
* Are any old or weak cryptographic algorithms or protocols used either by default or in older code?
* Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing? Are crypto keys checked into source code repositories?
* Is encryption not enforced, e.g., are any HTTP headers (browser) security directives or headers missing?
* Is the received server certificate and the trust chain properly validated?
* Are initialization vectors ignored, reused, or not generated sufficiently secure for the cryptographic mode of operation? Is an insecure mode of operation such as ECB in use? Is encryption used when authenticated encryption is more appropriate?
* Are passwords being used as cryptographic keys in absence of a password base key derivation function?
* Is randomness used for cryptographic purposes that was not designed to meet cryptographic requirements? Even if the correct function is chosen, does it need to be seeded by the developer, and if not, has the developer over-written the strong seeding functionality built into it with a seed that lacks sufficient entropy/unpredictability?
* Are deprecated hash functions such as MD5 or SHA1 in use, or are non-cryptographic hash functions used when cryptographic hash functions are needed?
* Are deprecated cryptographic padding methods such as PCKS number 1 v1.5 in use?
* Are cryptographic error messages or side channel information exploitable, for example in the form of padding oracle attacks?.

List of A02:2021 – Cryptographic Failures vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 4: A02:2021 – Cryptographic Failures Vulnerabilities*

## OWASP A03:2021 – Injection

An application is vulnerable to attack when:

* User-supplied data is not validated, filtered, or sanitized by the application.
* Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
* Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records.
* Hostile data is directly used or concatenated. The SQL or command contains the structure and malicious data in dynamic queries, commands, or stored procedures.

Some of the more common injections are SQL, NoSQL, OS command, Object Relational Mapping (ORM), LDAP, and Expression Language (EL) or Object Graph Navigation Library (OGNL) injection. The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections. Automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs is strongly encouraged.

List of A3-Sensitive Data Exposure vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 5: A03:2021 – Injection Vulnerabilities*

## OWASP A04:2021 – Insecure Design

Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” Insecure design is not the source for all other Top 10 risk categories. There is a difference between insecure design and insecure implementation. We differentiate between design flaws and implementation defects for a reason, they have different root causes and remediation. A secure design can still have implementation defects leading to vulnerabilities that may be exploited. An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.

List of A04:2021 – Insecure Design vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 6: A04:2021 – Insecure Design vulnerabilities*

## OWASP A05:2021 – Security Misconfiguration

The application might be vulnerable if the application is:

* Missing appropriate security hardening across any part of the application stack or improperly configured permissions on cloud services.
* Unnecessary features are enabled or installed (e.g., unnecessary ports, services, pages, accounts, or privileges).
* Default accounts and their passwords are still enabled and unchanged.
* Error handling reveals stack traces or other overly informative error messages to users.
* For upgraded systems, the latest security features are disabled or not configured securely.
* The security settings in the application servers, application frameworks (e.g., Struts, Spring, ASP.NET), libraries, databases, etc., are not set to secure values.
* The server does not send security headers or directives, or they are not set to secure values.
* The software is out of date or vulnerable (see A06:2021-Vulnerable and Outdated Components).

List of A05:2021 – Security Misconfiguration vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 7: A05:2021 – Security Misconfiguration vulnerabilities*

## OWASP A06:2021 – Vulnerable and Outdated Components

You are likely vulnerable:

* If you do not know the versions of all components you use (both client-side and server-side). This includes components you directly use as well as nested dependencies.
* If the software is vulnerable, unsupported, or out of date. This includes the OS, web/application server, database management system (DBMS), applications, APIs and all components, runtime environments, and libraries.
* If you do not scan for vulnerabilities regularly and subscribe to security bulletins related to the components you use.
* If you do not fix or upgrade the underlying platform, frameworks, and dependencies in a risk-based, timely fashion. This commonly happens in environments when patching is a monthly or quarterly task under change control, leaving organizations open to days or months of unnecessary exposure to fixed vulnerabilities.
* If software developers do not test the compatibility of updated, upgraded, or patched libraries.
* If you do not secure the components’ configurations (see A05:2021-Security Misconfiguration).

List of A06:2021 – Vulnerable and Outdated Components vulnerabilities that had any findings in this application.

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 8: A06:2021 – Vulnerable and Outdated Components vulnerabilities*

## OWASP A07:2021 – Identification and Authentication Failures

Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks. There may be authentication weaknesses if the application:

* Permits automated attacks such as credential stuffing, where the attacker has a list of valid usernames and passwords.
* Permits brute force or other automated attacks.
* Permits default, weak, or well-known passwords, such as "Password1" or "admin/admin".
* Uses weak or ineffective credential recovery and forgot-password processes, such as "knowledge-based answers," which cannot be made safe.
* Uses plain text, encrypted, or weakly hashed passwords data stores (see A02:2021-Cryptographic Failures).
* Has missing or ineffective multi-factor authentication.
* Exposes session identifier in the URL.
* Reuse session identifier after successful login.
* Does not correctly invalidate Session IDs. User sessions or authentication tokens (mainly single sign-on (SSO) tokens) aren't properly invalidated during logout or a period of inactivity.

List of A07:2021 – Identification and Authentication Failures vulnerabilities that had any findings in this application

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 9: A07:2021 – Identification and Authentication Failures vulnerabilities*

## OWASP A08:2021 – Software and Data Integrity Failures

Software and data integrity failures relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise. Lastly, many applications now include auto-update functionality, where updates are downloaded without sufficient integrity verification and applied to the previously trusted application. Attackers could potentially upload their own updates to be distributed and run on all installations. Another example is where objects or data are encoded or serialized into a structure that an attacker can see and modify is vulnerable to insecure deserialization.

List of A08:2021 – Software and Data Integrity Failures rules that had any findings in this application

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 10: A08:2021 – Software and Data Integrity Failures vulnerabilities*

## OWASP A09:2021 – Security Logging and Monitoring Failures

This category is to help detect, escalate, and respond to active breaches. Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time:

* Auditable events, such as logins, failed logins, and high-value transactions, are not logged.
* Warnings and errors generate no, inadequate, or unclear log messages.
* Logs of applications and APIs are not monitored for suspicious activity.
* Logs are only stored locally.
* Appropriate alerting thresholds and response escalation processes are not in place or effective.
* Penetration testing and scans by dynamic application security testing (DAST) tools (such as OWASP ZAP) do not trigger alerts.
* The application cannot detect, escalate, or alert for active attacks in real-time or near real-time.

List of A09:2021 – Security Logging and Monitoring Failures rules that had any findings in this application -

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 10: A09:2021 – Security Logging and Monitoring Failures vulnerabilities*

## OWASP A10:2021 – Server-Side Request Forgery (SSRF)

SSRF flaws occur whenever a web application is fetching a remote resource without validating the user-supplied URL. It allows an attacker to coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network access control list (ACL).

As modern web applications provide end-users with convenient features, fetching a URL becomes a common scenario. As a result, the incidence of SSRF is increasing. Also, the severity of SSRF is becoming higher due to cloud services and the complexity of architectures.

List of A10:2021 – Server-Side Request Forgery (SSRF) rules that had any findings in this application -

|  |  |  |  |
| --- | --- | --- | --- |
| CAST Rules | Total Vulnerabilities | Added Vulnerabilities | Removed Vulnerabilities |
| Rule 1 | 0 | 0 | 0 |
| Rule 2 | 0 | 0 | 0 |
| Rule 3 | 0 | 0 | 0 |
| Rule 4 | 0 | 0 | 0 |
| Rule 5 | 0 | 0 | 0 |

*Table 12: A10:2021 – Server-Side Request Forgery (SSRF) vulnerabilities*

# Appendix

## About CAST Software Intelligence

Software Intelligence creates understanding into software architecture, end to end transaction flows, data access patterns and more, helping teams work confidently and faster. Hundreds of companies rely on CAST Software Intelligence to improve end-user satisfaction and time-to-market, prevent business disruption and reduce cost, enabling them to move past today’s obstacles and to tackle the next wave of innovation.

[Click here](https://www.castsoftware.com/software-intelligence) for more information about CAST Software Intelligence.

## About CAST Security

Cyber risk and application security require a proactive and intelligence-driven approach. CAST Software Intelligence shifts insight into security strategy blind spots before development starts. With its unique ability to do dataflow and system-level analysis, CAST provides the most accurate security findings, reducing a lot of false positives. CAST Security rules are adapted from best-in-class industry standards – ISO, CWE, OWASP, PCI DSS, STIG, NIST, ….

To find out more about CAST Security, [click here](https://www.castsoftware.com/use-cases/application-security).