Acunetix
by Invicti

Compliance Report

OWASP TOP 10 2021

Description

The primary aim of the OWASP Top 10 is to educate developers, designers, architects, managers, and organizations about the consequences of the most important web application security weaknesses. The Top 10 provides basic techniques to protect against these high risk problem areas - and also provides guidance on where to go from here.

Disclaimer

Scan Detail

Target	https://localhost:7056/
Scan Type	Full Scan
Start Time	Jun 4, 2022, 3:46:44 PM GMT-7
Scan Duration	9 minutes
Requests	37328
Average Response Time	1ms
Maximum Response Time	2056ms

Compliance at a Glance

	CATEGORY
0	A01 Broken Access Control
0	A02 Cryptographic Failures
0	A03 Injection
0	A04 Insecure Design
0	A05 Security Misconfiguration
0	A06 Vulnerable and Outdated Components
0	A07 Identification and Authentication Failures
0	A08 Software and Data Integrity Failures
0	A09 Security Logging and Monitoring Failures
0	A10 Server-Side Request Forgery

Detailed Compliance Report by Category

This section is a detailed report that explains each vulnerability found according to individual compliance categories.

A01 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.

No alerts in this category

A02 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).

No alerts in this category

A03 Injection

Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

No alerts in this category

A04 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.

No alerts in this category

A05 Security Misconfiguration

Security misconfiguration is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securly configured, but they must be patched and upgraded in a timely fashion.

No alerts in this category

A06 Vulnerable and Outdated Components

Components, such as libraries, frameworks, and other software modules, almost always run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

No alerts in this category

A07 Identification and Authentication Failures

Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

No alerts in this category

A08 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.

No alerts in this category

A09 Security Logging and Monitoring Failures

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systesm, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

No alerts in this category

A10 Server-Side Request Forgery

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).

No alerts in this category

Coverage

- https://localhost:7056
  - _framework
    - aspnetcore-browser-refresh.js
    - blazor-hotreload
    - clear-browser-cache
  - swagger
    - v1
      - Currencies
        - fiat
      - Notification
        - bitpay
          - Inputs
            - POST url, status, currency, invoiceTime, expirationTime, currentTime, id, transactionCurrency, exceptionStatus, paymentCodes.bch.biP72b, paymentCodes.bch.biP73, paymentCodes.btc.biP72b, paymentCodes.btc.biP73, paymentCodes.busd.eiP681b, paymentCodes.dai.eiP681b, paymentCodes.doge.biP72b, paymentCodes.doge.biP73, paymentCodes.eth.eiP681, paymentCodes.gusd.eiP681b, paymentCodes.pax.eiP681b, paymentCodes.usdc.eiP681b, paymentCodes.wbtc.eiP681b, paymentCodes.xrp.biP72b, paymentCodes.xrp.biP73, paymentCodes.xrp.riP681, paymentDisplayTotals.btc, paymentDisplayTotals.bch, paymentDisplayTotals.eth, paymentDisplayTotals.gusd, paymentDisplayTotals.pax, paymentDisplayTotals.busd, paymentDisplayTotals.usdc, paymentDisplayTotals.xrp, paymentDisplayTotals.doge, paymentDisplayTotals.dai, paymentDisplayTotals.wbtc
        - coinbase
          - Inputs
            - POST data.id, data.created_at, data.expires_at, data.addresses.bitcoin, data.addresses.ethereum, data.pricing.bitcoin.amount, data.pricing.bitcoin.currency, data.pricing.ethereum.amount, data.pricing.ethereum.currency, data.pricing.local.amount, data.pricing.local.currency, data.timeline.1.time, data.timeline.1.status, data.timeline.1.context
        - coinpayments
          - Inputs
            - POST txn_id, amount1, amount2, currency1, currency2, status, status_text
        - coinqvest
          - Inputs
            - POST eventType, data.checkout.id, data.checkout.state
      - Payment

- Payment
  - Inputs
    - POST amount, fiatCurrency, cryptoCurrency, transactionReference
- swagger.json
  - Inputs
    - GET password, username
- swagger.yaml

- index.html
  - #fragments
    - javascript:domxssExecutionSink(1,javascript:domxssExecutionSink(1,"'/"%3E%3Cxsstag%3E()hashxss")
  - Inputs
    - GET password, username
- swagger-ui-bundle.js
- swagger-ui-standalone-preset.js
- swagger-ui.css