APPTRANA™

Indusface TAS QA
http://abrarhybrid4.qa-apptrana.info
2024-11-12

API Security Audit Report

Scan Summary

This site was checked through our Indusface WAS automated scannerand & Manual PT team, we have found 28 number of vulnerabilities.

Purpose and Scope

As part of assessment Indusface conducted web application security automated scanning with valid authentication & manual penetration testing for Indusface TAS QA on http://abrarhybrid4.qa-apptrana.info. Assessment scope includes

Assessment scope includes

OWASP 1

SQL Injection
LDAP Injection
Command Injection
SSI Injection
Code Injection

OWASP 2

Cookie Attributes
Privilege Escalation
Exposed Session Variable

OWASP 3

Reflected Cross Site Scripting
Stored Cross Site Scripting
Dom based CSS Scripting
HTML Injection
XSS Filter Evasion

OWASP 4

Directory Traversal
Insecure Direct Object Reference
Local File Inclusion

OWASP 5

Fingerprint Web Server, Web Application Framework,Web Applications
Enumerate Infrastructure and Application Admin Interface
HTTP Methods, Error Codes, Stack Traces IA Cross Domain Policy

OWASP 6

Insecure cryptographic storage
Insufficient transport layer protection.
Check for SSL certificate attributes

OWASP 7

OWASP 8

Cross-Site request forgery

OWASP 9

Known vulnerable framework and Library
Check for SSL certificate attributes

OWASP 10

Invalidated redirects and forwards
check for redirection to blacklisted URL

LOGICAL CHECKS

- Abuse of Functionality
- Insufficient Anti-automation
- Insufficient Authentication
- Insufficient process validation
- Server side validation
- Email ids can be harvested for spamming
- By Pass Authentication
- Insufficient password recovery
- Application does not display last login time

INDUSFACE DEFINED CHECKS

- Check for Encoding
- Password Auto-Complete
- Improper implementation of SSL (cipher, version)
- Service enumeration
- Port scanning
- Hidden iframe detection
- Malicious file can be uploaded on the server
- Steal password from browser memory
- Check HTTP methods
- Check for cookie attributes

Application specific business logic checks

Application specific business logical checks were also carried out by security experts. These experts went through a walkthrough of the web application to understand various threat scenarios and functionality of the application before a business logic test was performed.

Methodology

The testing process includes a series of steps, each of which are meant to provide the tester additional knowledge of application and to identify the existence of a vulnerability. The application security assessment is done in following four steps,

Phase 1 | Reconnaissance

Reconnaissance is the act of gathering preliminary data or intelligence on target. The data is gathered in-order to better plan the assessment process. Reconnaissance can be performed actively (meaning, directly touching the target) or passively (meaning that recon is being performed through an intermediary).

Phase 2 | Configuration Management

In this step, the application contents are discovered, security architecture of application is reviewed, cross domain policy is verified, URL and sensitive data at client-side is verified. Testing for Secure transmission of data is performed in this stage.

Phase 3 | Authentication and Session Management

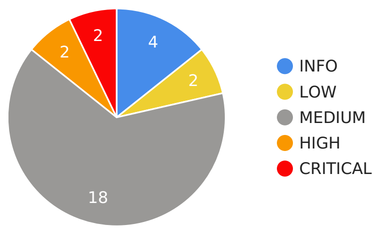
This phase validates the Application password functionality, CAPTCHA, Multi factor authentication, default logins, weak security questions and Session enumeration, session replay, session termination, session randomness, session authentication, hijacking of session cookie, CSRF and clickjacking is tested.

Phase 4 | Authorization and Data Validation

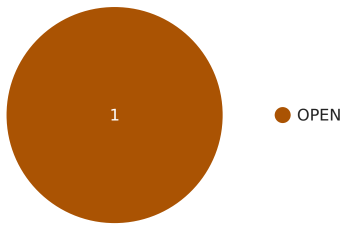
The results of this phase give results of Path traversal, vertical and horizontal privilege escalation, IDOR, weak algorithms, HTML Injection, XPATH, Command, SQL, LDAP, XML, XXI injections and results for Cross site scripting, URL Re-direction, File inclusion and data integrity.

Scan Overview

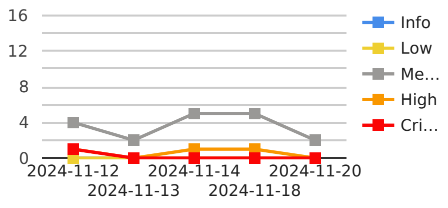
Number of Vulnerability



Progress from Last scan on 2024-11-12



Trend Over Last 5 scans



Vulnerability	Severity	Count

SQL Injection	Critical	2
Missing Account Lockout Policy	High	1
SQL Injection Authentication Bypass	High	1
Application Error Message	Medium	9
Content Injection	Medium	2
Credit Card Number Disclosure	Medium	1
Improper Token Handling	Medium	5
Sensitive Form Data Submitted In Cleartext	Medium	1
Possible Slow Response Time Detected	Low	1
Unencoded special characters	Low	1
Email Address Disclosure	Info	1
Reveals Sensitive Information	Info	2
Running Service(Open Port)	Info	1

OWASP Summary

OWASP Category	Vulnerability Title	No Of Vul Found
A2: Cryptographic Failures	Reveals Sensitive Information	2
A2: Cryptographic Failures	Unencoded special characters	1
A3: Injection	SQL Injection	2
A3: Injection	Content Injection	2
A4: Insecure Design	Possible Slow Response Time Detected	1
A5: Security Misconfiguration	Running Service(Open Port)	1
A5: Security Misconfiguration	Application Error Message	9
A5: Security Misconfiguration	Credit Card Number Disclosure	1
A5: Security Misconfiguration	Email Address Disclosure	1
A7: Identification and Authentication Failures	Sensitive Form Data Submitted In Cleartext	1
A7: Identification and Authentication Failures	SQL Injection Authentication Bypass	1
A7: Identification and Authentication Failures	Improper Token Handling	5
A7: Identification and Authentication Failures	Missing Account Lockout Policy	1

Vulnerability: SQL Injection

URL: http://
Severity: Critical — Found On: 2024-11-12
Method: POST — Status: NEW
OWASP Category: A3: Injection

Description: Web applications that do not properly sanitize user input before passing it to a database system are vulnerable to SQL injection. This type of attack potentially allows a malicious user to recover and/or modify any data that the application has access to.

How to fix: SQL Injection can be protected by, 1.sanitizing all user-submitted data via input validate functions to defend common source of SQL Injection. 2.using prepared statements with parameterized queries as a strong control to mitigate SQL attacks by ensuring the SQL interpreter always differentiates between code and data. 3.implementing stored procedures safely by avoiding dynamic SQL generation inside. 4.using appropriate privileges and follow

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:al

Response: HTTP/1.1 400
content-type:text\/plain
content-length:68
date:Tue, 12 Nov 2024 13:40:09 GMT
connection:close

Result: Syntax error: Encountered

Proof of Concept:

Vulnerability: SQL Injection

URL: http:/

Severity: Critical — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A3: Injection

Description: Web applications that do not properly sanitize user input before passing it to a database system are vulnerable to SQL injection. This type of attack potentially allows a malicious user to recover and/or modify any data that the application has access to.

How to fix: SQL Injection can be protected by, 1.sanitizing all user-submitted data via input validate functions to defend common source of SQL Injection. 2.using prepared statements with parameterized queries as a strong control to mitigate SQL attacks by ensuring the SQL interpreter always differentiates between code and data. 3.implementing stored procedures safely by avoiding dynamic SQL generation inside. 4.using appropriate privileges and follow

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 400
content-type:text\/plain
content-length:73
date:Tue, 12 Nov 2024 13:41:02 GMT
connection:close

Result: Syntax error: Encountered

Proof of Concept:

Vulnerability: SQL Injection Authentication Bypass

URL: http://

Severity: High — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: Web applications with weak authentication controls & access control polices may allow remote attackers to bypass authentication by injecting crafted SQL queries during login attempts. Successful attacks result in unauthenticated, remote attackers to gain complete control of the account/admin privileges and conducts attacks further.

How to fix: It is suggested to enforce strong access policies/authentication controls/methods including anti brute force and session protection mechanisms. Enforce user input validation to ensure input values are not misused. For ex: Not allowing special chars, invalid length, etc. for username & password field values.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host

Response: HTTP/1.1 200
content-type:application\/json
content-length:112
date:Tue, 12 Nov 2024 13:40:09 GMT
connection:close

Result: Website is vulnerable to authentication bypass attack with below payloads in username and password: username: 'or'1'='1 password: Original Vaule

Proof of Concept:

Vulnerability: Missing Account Lockout Policy

URL:

Severity: High — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: Multiple unsuccessful login attempts with invalid passwords is suspicious behaviour as it may be caused by brute force password guessing attacks which are intend to steal sensitive information, get access to administrative panels to perform unauthorized transactions or assisting to perform further attacks. To mitigate this issue, account lockout mechanisms are used and such locked out accounts can only be unlocked after a predetermined period of time, via a self-service unlock mechanism, or intervention by an administrator.

How to fix: '- Ensure to have strong account lockout policy and set the invalid login attempts limit to acceptable limit. - Strengthen Password policy to avoid usage of weak passwords. - Implement multi-factor authentication to avoid unauthorized access when valid credentials are obtained - Introduce CAPTCHA after a failed login attempt.'

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 400
content-type:text\/plain
content-length:83
date:Tue, 12 Nov 2024 13:47:11 GMT
connection:close

Result: Website is having no account lockout mechanism when attempted to login with invalid credentials few times: http://abrarhybrid4.qa-apptrana.info/api/login

Proof of Concept:

Vulnerability: Sensitive Form Data Submitted In Cleartext

URL: http://

Severity: Medium Found On: 2024-11-12

Method: POST Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: A web form contains fields with data that is probably sensitive in nature. This form data is submitted over an unencrypted connection, which could allow hackers to sniff the network and view the data in plaintext.

How to fix: Use the HTTPS (HTTP over SSL) protocol to submit sensitive form data Enable the HTTPS protocol on the server. Change the "action" URL of the form tag to use the HTTPS protocol ("https://...") instead of just the HTTP protocol ("http://..."). All sensitive data should be sent over HTTPS instead of over HTTP.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:110
date:Tue, 12 Nov 2024 13:39:47 GMT
connection:close

Result: Vulnerabilty Found

Proof of Concept:

Vulnerability: Application Error Message

URL: http://

Severity: Medium Found On: 2024-11-12

Method: POST Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 13:40:28 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Application Error Message

URL: http://
Severity: Medium
Found On: 2024-11-12
Method: GET
Status: NEW
OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: GET /api/account/800002/transactions HTTP/1.1
Authorization:Bearer YW5OdGFYUm86WkdWdGl6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 13:42:34 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Improper Token Handling

URL: http:/
Severity: Medium
Found On: 2024-11-12
Method: POST
Status: NEW
OWASP Category: A7: Identification and Authentication Failures

Description: Token based transactions are used to store the user state on the client and the user data is encrypted into a token with a secret and then sent back to the client. Improper handling of tokens like not generating unique per user session or improper validation before accepting and executing it may allow remote attackers to bypass authentication mechanisms, CSRF etc.

How to fix: Tokens should be unique with respect to every user session and shouldn’t be reused. The server should reject the requested action if the token fails validation. Token should not be pass in request URL.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Referer:& ping -n 3 127.0.0.1 &
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:101
date:Tue, 12 Nov 2024 13:44:44 GMT
connection:close

Result: Sensitive data disclosed: Valid token not found

Proof of Concept:

Vulnerability: Improper Token Handling

URL: http:/
Severity: Medium
Found On: 2024-11-12
Method: POST
Status: NEW
OWASP Category: A7: Identification and Authentication Failures

Description: Token based transactions are used to store the user state on the client and the user data is encrypted into a token with a secret and then sent back to the client. Improper handling of tokens like not generating unique per user session or improper validation before accepting and executing it may allow remote attackers to bypass authentication mechanisms, CSRF etc.

How to fix: Tokens should be unique with respect to every user session and shouldn’t be reused. The server should reject the requested action if the token fails validation. Token should not be pass in request URL.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
User-Agent:'+convert(varchar,0x7b5d)+'
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:106
date:Tue, 12 Nov 2024 14:01:53 GMT
connection:close

Result: Sensitive data disclosed: Valid token not found

Proof of Concept:

Vulnerability: Improper Token Handling

URL: http://

Severity: Medium | Found On: 2024-11-12

Method: POST | Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: Token based transactions are used to store the user state on the client and the user data is encrypted into a token with a secret and then sent back to the client. Improper handling of tokens like not generating unique per user session or improper validation before accepting and executing it may allow remote attackers to bypass authentication mechanisms, CSRF etc.

How to fix: Tokens should be unique with respect to every user session and shouldn’t be reused. The server should reject the requested action if the token fails validation. Token should not be pass in request URL.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Origin:\" location.replace('http:\/\/0b676960-1902-4ca8-8368-740a99cc0ffe.804.16323.665.332836.haikuscan.indusfacefinder.in\/')
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:109
date:Tue, 12 Nov 2024 14:03:50 GMT
connection:close

Result: Sensitive data disclosed: Valid token not found

Proof of Concept:

Vulnerability: Application Error Message

URL: http://

Severity: Medium | Found On: 2024-11-12

Method: POST | Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: POST /api/account/800002/transactions HTTP/1.1
Content-Type:application\/json
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:a

Response: HTTP/1.1 500
content-type:text\/html;charset=utf-8
content-language:en
content-length:1816
date:Tue, 12 Nov 2024 14:07:29 GMT
connection:close

Result: ;} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 500 – Internal Server Error</h1><hr class="line" /><p><b>Type</b> Exception Report</p><p><b>Message</b> java.lang.NullPointerException</p><p><b>Description</b> The server encountered an unexpected condition that prevented it from fulfilling the request.</p><p><b>Exception</b></p><pre>javax.servlet.ServletException: java.lang.NullPoi

Proof of Concept:

Vulnerability: Credit Card Number Disclosure

URL: http://a

Severity: Medium — Found On: 2024-11-12

Method: GET — Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: This is intended to detect sensitive and financial information such as Credit Card Number in HTTP responses. Credit Card Number disclosure may allow remote attackers to steal other financial information and make unknown transactions.

How to fix: It is advised not to show any critical financial/sensitive information on website or pages publicly available. Showing the info should go through proper authentication and authorization.

Request: GET /api/account HTTP/1.1
Authorization:Bearer YW5OdGFYUm86WkdWdGI6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:148
date:Tue, 12 Nov 2024 14:07:28 GMT
connection:close

Result: Following credit card numbers found: Visa : "4539082039396288",4539082039396288

Proof of Concept:

Vulnerability: Application Error Message

URL: http://

Severity: Medium — Found On: 2024-11-12

Method: GET — Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: GET /api/account HTTP/1.1
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 14:08:48 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Improper Token Handling

URL: http://a

Severity: Medium — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: Token based transactions are used to store the user state on the client and the user data is encrypted into a token with a secret and then sent back to the client. Improper handling of tokens like not generating unique per user session or improper validation before accepting and executing it may allow remote attackers to bypass authentication mechanisms, CSRF etc.

How to fix: Tokens should be unique with respect to every user session and shouldn't be reused. The server should reject the requested action if the token fails validation. Token should not be pass in request URL.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:146
date:Tue, 12 Nov 2024 14:20:36 GMT
connection:close

Result: Sensitive data disclosed: Valid token not found

Proof of Concept:

Vulnerability: Improper Token Handling

URL: http://[redacted]

Severity: Medium — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A7: Identification and Authentication Failures

Description: Token based transactions are used to store the user state on the client and the user data is encrypted into a token with a secret and then sent back to the client. Improper handling of tokens like not generating unique per user session or improper validation before accepting and executing it may allow remote attackers to bypass authentication mechanisms, CSRF etc.

How to fix: Tokens should be unique with respect to every user session and shouldn’t be reused. The server should reject the requested action if the token fails validation. Token should not be pass in request URL.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host: [redacted]

Response: HTTP/1.1 200
content-type:application\/json
content-length:136
date:Tue, 12 Nov 2024 14:20:13 GMT
connection:close

Result: Sensitive data disclosed: Valid token not found

Proof of Concept:

Vulnerability: Content Injection

URL: http://[redacted]

Severity: Medium — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A3: Injection

Description: Content spoofing, also referred to as content injection or context injection, is an attack targeting a user made possible by an injection vulnerability in a web application. When an application does not properly handle user supplied data, an attacker can supply content to a web application, typically via a parameter value, that is reflected back to the user. This presents the user with a modified page under the context of the trusted domain. This attack is typically used as, or in conjunction with, social engineering because the attack is exploiting a code-based vulnerability and a user's trust.

How to fix: 1) User supplied input should not be directly reflected. 2) Implement Custom error page.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host: [redacted]

Response: HTTP/1.1 200
content-type:application\/json
content-length:205
date:Tue, 12 Nov 2024 14:20:39 GMT
connection:close

Result: Text Injected: Site is moved to wastest.indusface.com kindly visit wastest.indusface.com.

Proof of Concept:

Vulnerability: Application Error Message

URL: http://[redacted]

Severity: Medium — Found On: 2024-11-12

Method: GET — Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: GET /api/feedback/123 HTTP/1.1
Content-Type:application\/json

Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:

Response: HTTP/1.1 500
content-type:text\/html;charset=utf-8
content-language:en
content-length:4460
date:Tue, 12 Nov 2024 14:20:35 GMT
connection:close

Result: lang.IndexOutOfBoundsException: Index: 0, Size: 0</p><p><b>Description</b> The server encountered an unexpected condition that prevented it from fulfilling the request.</p><p><b>Exception</b></p><pre>javax.servlet.ServletException: java.lang.IndexOutOfBoundsException: Index: 0, Size: 0
org.glassfish.jersey.servlet.WebComponent.serviceImpl(WebComponent.java:432)
org.glassfish.jersey.servlet.WebComponent.service(WebComponent.j

Proof of Concept:

Vulnerability: Application Error Message

URL: http:/,

Severity: Medium — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 14:20:57 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Application Error Message

URL: http://

Severity: Medium — Found On: 2024-11-12

Method: POST — Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: POST /api/admin/addUser HTTP/1.1
Content-Type:application\/json
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 14:33:37 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Application Error Message

URL: http://

Severity: Medium

Found On: 2024-11-12

Method: POST

Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error [illegible] s kind of vulnerabilities.

Request: POST /api/admin/changePassword HTTP/1.1
Content-Type:application\/json
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host: [illegible]

Response: HTTP/1.1 400
content-type:text\/html;charset=utf-8
content-language:en
content-length:1127
date:Tue, 12 Nov 2024 14:34:24 GMT
connection:close

Result: process the request due to something that is perceived to be a client error (e.g., malformed request syntax, invalid request message framing, or deceptive request routing).</p><hr class="line" /><h3>Apache Tomcat/8.5.43</h3></body></html>

Proof of Concept:

Vulnerability: Application Error Message

URL: http:// [illegible]

Severity: Medium

Found On: 2024-11-12

Method: POST

Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An attacker can try to force the target website to produce error messages by passing different attack vectors to different parameters and then analyse the errors to get target information. These errors have no direct security impact, most of the time they indicate a programming error, quality issue, or a potential vulnerability in the application. Many of these types of errors also leak information about the logic or the implementation of the application which can help an attacker to identify or exploit weaknesses in the application.

How to fix: Configure your server or application to log the error rather showing the error on the web page Input validation and proper error handling is the best approach to overcome this kind of vulnerabilities.

Request: POST /api/transfer HTTP/1.1
Content-Type:application\/json
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host: [illegible]

Response: HTTP/1.1 500
content-type:text\/html;charset=utf-8
content-language:en
content-length:4533
date:Tue, 12 Nov 2024 14:47:58 GMT
connection:close

Result: ;} .line {height:1px;background-color:#525D76;border:none;}</style></head><body><h1>HTTP Status 500 – Internal Server Error</h1><hr class="line" /><p><b>Type</b> Exception Report</p><p><b>Message</b> java.lang.NumberFormatException: For input string: &quot;&#39; AND SLEEP(0)=&#39; &quot;</p><p><b>Description</b> The server encountered an unexpected condition that prevented it from fulfilling the request.</p><p><b>Exception</b></p> <pre>javax.servl

Proof of Concept:

Vulnerability: Content Injection

URL: http://

Severity: Medium

Found On: 2024-11-12

Method: POST

Status: NEW

OWASP Category: A3: Injection

Description: Content spoofing, also referred to as content injection or context injection, is an attack targeting a user made possible by an injection vulnerability in a web application. When an application does not properly handle user supplied data, an attacker can supply content to a web application, typically via a parameter value, that is reflected back to the user. This presents the user with a modified page under the context of the trusted domain. This attack is typically used as, or in conjunction with, social engineering because the attack is exploiting a code-based vulnerability and a user's trust.

How to fix: 1) User supplied input should not be directly reflected. 2) Implement Custom error page.

Request: POST /api/transfer HTTP/1.1
Content-Type:application\/json
Authorization:Bearer YW5OdGFYUm86WkdWdGJ6RXlNelE9OmLvv70b77+9bO+\/ve+\/vQ==
Host:

Response: HTTP/1.1 500
content-type:text\/html;charset=utf-8
content-language:en
content-length:4686
date:Tue, 12 Nov 2024 14:48:00 GMT
connection:close

Result: Text Injected: p><b>Type</b> Exception Report</p><p><b>Message</b> java.lang.NumberFormatException: For input string: &quot; Site is moved to wastest.indusface.com kindly visit wastest.indusface.com. &quot;</p

Proof of Concept:

Vulnerability: Possible Slow Response Time Detected

URL: http://

Severity: Low	Found On: 2024-11-12
Method: POST	Status: NEW

OWASP Category: A4: Insecure Design

Description: Server response time is the amount of time required to load the HTML page of an application from a server so that the client (browser) can begin rendering the page. Without a good server response time, the HTML page will take longer to load. If the HTML page is not loaded, then browser won't know what other resources will be required in order to display the page properly. Web pages with slow response time can be targeted to be used in conducting DOS attacks to overload the servers and may result in an unresponsive application.

How to fix: Investigate and resolve issues which affects Server response time. Server response time over 200ms on an average is considered to be suspicious.

Request: POST /;'or HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:text\/html;charset=ISO-8859-1
transfer-encoding:chunked
date:Tue, 12 Nov 2024 13:54:21 GMT
connection:close

Result: Response time is : 5412.722805999219 and First Byte Response time is: 1211.203224003315

Proof of Concept:

Vulnerability: Unencoded special characters

URL: http://

Severity: Low	Found On: 2024-11-12
Method: POST	Status: NEW

OWASP Category: A2: Cryptographic Failures

Description: Unencoded characters is deficiency or bug which allows user to inject unsafe characters which alters HTML output and can generate other security vulnerabilities like XSS and HTML injection.

How to fix: All unsafe characters must always be encoded.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:147
date:Tue, 12 Nov 2024 14:20:38 GMT
connection:close

Result: The following unencoded characters are found: <xss=haikutest(1)>

Proof of Concept:

Vulnerability: Running Service(Open Port)

URL: http:/

Severity: Info | Found On: 2024-11-12

Method: POST | Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: An open port is possible leading to data loss, DOS attack and other vulnerabilities.

How to fix: Close unwanted port.

Request: POST /api/login HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 400
content-type:text\/plain
content-length:83
date:Tue, 12 Nov 2024 13:40:09 GMT
connection:close

Result: Found open port: 80

Proof of Concept:

Vulnerability: Reveals Sensitive Information

URL: http:/

Severity: Info | Found On: 2024-11-12

Method: POST | Status: NEW

OWASP Category: A2: Cryptographic Failures

Description: Sensitive information in Request and Response should be encoded with proper technique with salting. eg: Password, Account Details, Personal Identity information, etc. This may lead to exposure of other vulnerabilities.

How to fix: Use proper encoding technique with salting for sensitive field.

Request: POST /manager HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 403
content-type:text\/html;charset=ISO-8859-1
content-length:3446
date:Tue, 12 Nov 2024 14:00:48 GMT
connection:close

Result: Sensitive data disclosed: username="tomcat" password="s3cret"

Proof of Concept:

Vulnerability: Email Address Disclosure

URL: http:/

Severity: Info | Found On: 2024-11-12

Method: POST | Status: NEW

OWASP Category: A5: Security Misconfiguration

Description: There are number of crawlers running across the Internet to search email addresses from all the publicly available websites. Such crawlers crate a mailing list to keep sending spam emails. If your email address (example: sales@yourwebsite.com) gets listed in one of such mailing list, your inbox will receive dozens of spam on a daily basis. This may lead to missing out an important email.

How to fix: Obfuscate the email address, or place it in a form of image file.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:136
date:Tue, 12 Nov 2024 14:20:13 GMT
connection:close

Result: Following email address found: jsmtih@altoromutual.com

Proof of Concept:

Vulnerability: Reveals Sensitive Information

URL: http://
Severity: Info
Method: POST
OWASP Category: A2: Cryptographic Failures

Found On: 2024-11-12
Status: NEW

Description: Sensitive information in Request and Response should be encoded with proper technique with salting. eg: Password, Account Details, Personal Identity information, etc. This may lead to exposure of other vulnerabilities.

How to fix: Use proper encoding technique with salting for sensitive field.

Request: POST /api/feedback/submit HTTP/1.1
Content-Type:application\/json
Host:

Response: HTTP/1.1 200
content-type:application\/json
content-length:217
date:Tue, 12 Nov 2024 14:25:25 GMT
connection:close

Result: Sensitive data disclosed: email":"system(\"

Proof of Concept:

Confidentiality

Disclaimer