INDUSFACE™

Indusface WAS Scanned Vulnerabilities

Disclaimer

This document has been prepared by Indusface for informational purposes. Neither this document nor its contents may be reproduced, copied, or distributed in any form without prior written approval from Indusface.

Notice of Ownership

Vulnerabilities Scanned

Below is the complete list of Indusface WAS scanned vulnerabilities:

Sr. No.	Title	Description	Severity
1	HTTP DELETE Method Enabled	HTTP 'DELETE' method allows a client to delete a file on the web server. An attacker can exploit it as a very simple and direct way to deface a web site or to mount a DoS attack.	Low
2	HTTP Response Splitting	HTTP response splitting is a form of web application attack where unsafe characters are inserted into user-controllable fields which are later inserted into the HTTP header being used for 302 redirects. As per RFC standard, HTTP request headers are separated by one carriage return and line feed, and response headers are separated by two carriage return (CR) and line feed (LF). The response splitting attack consists of making the server print a carriage return line feed sequence followed by content supplied by the attacker in the header section of its response, typically by including them in input fields sent to the application. Response splitting can be used to perform CRLF injection that allows the attacker to set fake cookies, steal CSRF tokens, disclose user information by injecting a script (XSS) and perform a variety of other attacks. It also allows attackers to deactivate & bypass security measures like XSS filters & Same Origin Policy (SOP).	High
3	Microsoft IIS Internal IP Address Disclosure (CVE-2002-0422)	Certain WebDAV methods, when requested with a blank Host field, will return the internal IP of the target host machine.This IP can be used in subsequent attacks to further exploit the target system.	Low
4	Source Code Disclosure	Source code disclosure allows a malicious user to obtain the source code of a server-side application from a webpage. The attacker can obtain deeper knowledge of the Web application logic . Disclosure of source code and configuration files can be devastating for a web application. They usually contain database connection information like IP address, port number and valid credentials. In certain cases, application test users	Medium
5	Cross-Site Scripting (XSS)	The Web application is vulnerable to cross-site scripting (XSS), which allows attackers to inject JavaScript or HTML code executed on the client-side browser.	High

6	Directory Listing	A web directory was found to be browsable, which means that anyone can see the contents of the directory.	Medium
7	SQL Injection	Web applications that do not properly sanitize user input before passing it to a database system are vulnerable to SQL injection.	Critical
8	TLS/SSL Server Certificate Expired	The server's HTTPS X.509 certificate is expired.	Critical
9	HTTP TRACE Method Enabled	The HTTP TRACE method allows an attacker to capture client cookies via a Cross-Site Scripting attack.	Low
10	Sensitive Form Data Submitted In Cleartext	A web form contains sensitive fields submitted over an unencrypted connection.	Medium
11	ASP.NET Debug Feature Enabled	The ASP.NET application is running in debug mode which leaks source code and detailed error messages.	Medium
12	HTTP PUT Method Enabled	The Web server may allow a remote attacker to upload arbitrary files using the HTTP 'PUT' method.	Low
13	Possible Physical Path Disclosure	The web page may disclose the physical path of the web root.	Medium
14	Missing Secure Flag From Cookie Header	The Secure attribute tells the browser to only send the cookie if the request is being sent over a secure channel such as HTTPS.	Low
15	Sensitive HTML Form Fields With auto-complete Enabled	The Web form contains passwords or sensitive fields for which browser auto-complete is enabled.	Low
16	HTTP Basic Authentication Enabled	The HTTP Basic Authentication scheme passes user name and password over the network as cleartext.	Medium
17	OS Command Injection	An OS command injection vulnerability occurs when invalidated user-controlled parameters are used to execute operating system commands.	Critical
18	Remote File Inclusion (RFI)	Malicious file execution vulnerabilities allow attackers to include hostile external content processed by the web server.	Critical
19	ASP.NET Unencrypted '__VIEWSTATE' Parameter	The application uses ASP.NET viewstate without encryption to maintain application state.	Medium

20	XPath Injection	XPath injection allows a malicious user to insert arbitrary XPath code to bypass authentication or access restricted XML data.	High
21	Missing HttpOnly Flag From Cookie	Missing HttpOnly flag allows client-side scripts to access the cookie, enabling XSS-based cookie theft.	Low
22	Unvalidated Redirects And Forwards/Open Redirection	An open redirect vulnerability allows attackers to redirect users to malicious sites via phishing emails.	Medium
23	Invalid TLS/SSL Server Certificate	The server's TLS/SSL certificate signature is invalid, possibly indicating an active eavesdropping attempt.	Critical
24	Untrusted TLS/SSL Server Certificate	The server's TLS/SSL certificate is signed by an untrusted CA, indicating a possible man-in-the-middle attack.	Critical
25	Application Error Message	Error messages may leak information about the application's logic or implementation.	Medium
26	Email Address Disclosure	Publicly disclosed email addresses can be harvested by spam crawlers.	Info
27	Password Field Submitted Using GET Method	The page contains a form submitting a password via GET, exposing it in the URL and browser history.	Critical
28	SQL Statement In HTML Comment	An SQL statement found in a webpage comment can assist attackers in exploiting database vulnerabilities.	Medium
29	Internal IP Address Disclosure	Internal IP data may be used by an attacker to exploit the target hosting network.	Low
30	Possible Backup File(s) Detected	Backup files found on the web server may expose source code or configuration.	Medium
31	Possible Sensitive Directories/Files Detected	Sensitive directories or files may expose information that helps attackers mount advanced attacks.	Medium
32	Local File Inclusion (LFI)	The script may include files whose names are determined by user-supplied data without proper validation.	High
33	Permissive Cross-domain Policy File Detected	Permissive crossdomain.xml policy files allow external scripts to interact with the website.	Low

34	Readable .htaccess File Detected	The htaccess configuration file is readable, potentially exposing server configuration.	Medium
35	TLS/SSL Server Certificate Will Expire Soon	SSL Certificate is about to expire, which will undermine trust and cause browser warnings.	High
36	Web Server Info Disclosure	HTTP web server information disclosed in HTTP headers may help attackers find version-specific vulnerabilities.	Info
37	Robots.txt File Detected	Robots.txt file displays information about site directory structure that may assist attackers.	Info
38	ASP.NET ViewState MAC Disabled	Disabled MAC on ViewState allows attackers to tamper with the viewstate data.	Low
39	Programming Language And Version Information Disclosure	Framework and version information in HTTP headers may help attackers find version-specific vulnerabilities.	Info
40	HTML Injection	HTML injection allows attackers to inject valid HTML code into an application, enabling further attacks.	High
41	Predictable Resource Location	Predictable resource paths can be brute-forced to discover hidden resources.	Info
42	Insecure Content Security Policy (CSP)/X-Frame-Options	Missing or misconfigured CSP/X-Frame-Options headers leave the site vulnerable to Clickjacking.	Medium
43	Session ID In URL	Storing session information in the URL exposes session tokens to theft via logs and referrer headers.	Medium
44	HTTP Host Header Injection	Unvalidated Host header values can lead to Cache Poisoning, XSS, and other vulnerabilities.	Medium
45	Unencoded special characters	Unencoded special characters allow injection of unsafe characters that can trigger XSS or HTML injection.	Low
46	Cross-Origin Resource Sharing (CORS)	Permissive CORS policy can allow malicious cross-domain interactions with the application.	Low
47	Missing Account Lockout Policy	Without account lockout, brute force attacks can be used to guess user credentials.	High

48	HTTP Verb Tampering	HTTP Verb Tampering allows an attacker to bypass authorization by manipulating HTTP methods.	Low
49	Old SSL/TLS Version Detected	Old SSL/TLS versions (SSLv3, TLSv1, TLSv1.1) are vulnerable to BEAST, POODLE, and similar attacks.	Medium
50	Database Error Message	Database error messages may disclose sensitive information usable in further attacks.	Medium
51	CVS Web Repository Disclosure	CVS Web Repository found may expose file listings, paths, and repository structure.	Medium
52	Web Admin Homepage (webadmin.php) Script	webadmin.php file manager has no authentication in its default configuration.	High
53	AWS Metadata Server Side Request Forgery	SSRF vulnerability allows an attacker to access AWS metadata including secret keys and tokens.	High
54	X-XSS-Protection Header Disabled	Disabled X-XSS-Protection header leaves sites at risk from browser-level XSS attacks.	Low
55	Suspicious HTML Comments Detected	HTML comments may disclose sensitive information like credentials or file locations.	Low
56	User Controllable HTML Attribute	Unvalidated user-controlled HTML attributes can cause XSS or HTML injection vulnerabilities.	Medium
57	Form Action Hijacking	Allows an attacker to hijack form action URLs, capturing form content including CSRF tokens.	Medium
58	Insecure Flash Parameter 'AllowScriptAccess' Detected	AllowScriptAccess set to 'always' allows arbitrary JavaScript execution via Flash objects.	Medium
59	Web Server Default Web Page Detected	Default web server pages disclose platform and version information.	Medium
60	HTTP OPTIONS Method Enabled	The OPTIONS method exposes a list of supported methods, potentially revealing sensitive server capabilities.	Low
61	SSL Certificate Common Name Mismatch	SSL certificate common name mismatch may indicate misconfiguration or an active MITM attack.	Critical

62	SSL Certificate Signed Using Weak Signature Algorithm	Certificates signed with MD2, MD4, MD5, or SHA1 are vulnerable to collision attacks.	Medium
63	SSL Certificate Using Weak Public Key	RSA keys less than 2048 bits are considered weak and increasingly vulnerable to brute force.	High
64	Apache Struts2 Development Mode Enabled	Development mode provides additional debug information and may allow arbitrary Java code execution.	Medium
65	Apache server-status Enabled	Server-status exposes uptime, request statistics, client IPs, and CPU usage to attackers.	Medium
66	ASP.NET Tracing Enabled	ASP.NET tracing exposes session IDs, execution paths, and other diagnostic information.	Medium
67	ASP.NET Version Disclosure	ASP.NET version in HTTP headers allows attackers to target version-specific vulnerabilities.	Info
68	Browser Cache Enabled	Cached web application data may expose URL histories, cookies, and transaction history.	Low
69	Hidden Form Input 'Price' Detected	Hidden form inputs can be manipulated by users to alter values like prices before form submission.	Medium
70	Possible Web Form Spam Detected	Poorly written form scripts may allow the application to be used as a spam relay.	Medium
71	Documentation File Detected	Documentation files like readme.txt may expose application name, version, and user details.	Low
72	Possible Slow Response Time Detected	Slow server response times can be exploited in DoS attacks to overload servers.	Low
73	Microsoft IIS Version Disclosure	IIS version in response headers allows attackers to target version-specific vulnerabilities.	Info
74	HTML Form Found In Redirect Page	An HTML form in a redirect page without response termination can bypass authentication.	Low
75	Server Side Request Forgery Local File Inclusion	SSRF vulnerability allows an attacker to perform local file inclusion, accessing sensitive files.	High

76	Unset/Insecure HSTS header	Missing HSTS header allows MITM attacks and theft of private data.	Medium
77	Cookie Scoped To Parent Domain	A cookie scoped to the parent domain is available to all subdomains, increasing leak risk.	Low
78	WordPress XML-RPC Interface Detected	XML-RPC interface may be used in brute force or amplification attacks.	Medium
79	Improper Token Handling	Improper token handling may allow authentication bypass and CSRF attacks.	Medium
80	Apache Tomcat Remote Code Execution Vulnerability (CVE-2019-0232)	CGI Servlet vulnerability on Windows allows unauthenticated remote code execution.	High
81	Credit Card Number Disclosure	Credit card numbers found in HTTP responses may allow financial data theft.	Medium
82	Oracle WebLogic Server Deserialization RCE (CVE-2019-2725)	Deserialization vulnerability in WebLogic allows unauthenticated remote command execution.	Critical
83	Dot Net Insecure Deserialization Remote Command Execution	Insecure deserialization of untrusted data allows DoS, authentication bypass, and RCE.	Critical
84	Perl Deserialization Remote Command Execution	Perl insecure deserialization allows authentication bypasses, DoS, and RCE.	High
85	Passive Mixed Content Vulnerability	HTTP content on HTTPS pages allows image replacement and page tampering.	Medium
86	Active Mixed Content Vulnerability	Mixed active content allows attackers to modify page behavior, steal credentials, or hijack sessions.	Medium
87	PHP Deserialization RCE Vulnerability (CVE-2017-17672)	PHP unserialize() abuse allows arbitrary file deletion or code execution.	High
88	Ruby on Rails XML/JSON Processor YAML Deserialization RCE (CVE-2013-0156)	RCE vulnerability in RoR XML processor allows arbitrary code execution.	High
89	Oracle WebLogic Deserialization RCE Bypass (CVE-2019-2729)	XMLDecoder deserialization flaw allows unauthenticated arbitrary code execution.	Critical

90	Possible Archive/Compression File(s) Detected	Archive files on the web server may expose source code or sensitive application data.	Low
91	Cookie Overly Broad Path Detected	Overly broad cookie path exposes cookies to other applications on the same domain.	Low
92	Session Cookie Manipulation	Misconfigured cookies can lead to XSS, SQL injection, or session fixation.	Medium
93	Weak Session IDs	Weak or predictable session IDs enable session hijacking attacks.	Medium
94	HTTP TRACK Method Enabled	HTTP TRACK method allows attackers to capture client cookies via a Cross-Site Scripting attack.	Low
95	Log Injection	Log injection allows attackers to insert malicious data into logs, corrupting records.	Medium
96	HTML Form Without Anti-CSRF Token Detected	Forms without CSRF tokens are vulnerable to Cross-Site Request Forgery attacks.	Medium
97	Web Administration Login Page Detected	Publicly accessible admin login pages may be targeted for brute-force or injection attacks.	Low
98	Web Server Content Sniffing Enabled	Content sniffing by browsers can be exploited to execute scripts as incorrect MIME types.	Medium
99	Apache Range Denial Of Service	Range header abuse in Apache allows a remote attacker to cause DoS via memory and CPU exhaustion.	Medium
100	VBulletin Pre-Auth RCE Vulnerability	Pre-authentication RCE in vBulletin allows attackers to execute commands and compromise systems.	Critical
101	Server Side Javascript Injection	User-controlled data processed by the server as JavaScript enables malicious code execution.	Critical
102	Server-Side Template Injection	User-controllable data in server-side templates allows arbitrary code execution.	Critical
103	Remote XSL Inclusion	Attacker-controlled XSL files can be included and executed to compromise the system.	High

104	PHP Nginx Remote Command Execution	PHP sites on Nginx servers are vulnerable to remote command execution.	Critical
105	Default And Common Credentials Detected	Common or default credentials allow attackers to gain full control of the application.	High
106	SQL Injection Authentication Bypass	SQL injection during login allows unauthenticated attackers to gain admin privileges.	High
107	Login Username Enumeration	Inconsistent error messages during login reveal valid usernames to attackers.	Medium
108	Core Dump File(s) Detected	Core dump files expose application memory including credentials and library data.	Medium
109	JSF Client-Side ViewState Detected	Unencrypted client-side ViewState can be read and used in further attacks.	Medium
110	WAF/IPS Detected	The site is protected by WAF/IPS; comprehensive vulnerability scanning may be limited.	Info
111	Insecure Cache-Control Header Detected	Misconfigured Cache-Control headers may cause sensitive pages to be served from cache.	Low
112	Old Cipher Suites Detected	Old cipher suites can be decoded to reveal sensitive information and enable SWEET32-type attacks.	Medium
113	Insecure/Deprecated Cryptography Detected	Weak or deprecated hashing/crypto functions can be decrypted to obtain sensitive information.	Medium
114	Apache Axis2 Local File Inclusion	LFI in Apache Axis2 allows attackers to access arbitrary files via crafted requests.	High
115	JSMOL2 Server Side Request Forgery Local File Inclusion	SSRF/LFI in JSMOL2 allows access to sensitive files like database credentials.	High
116	Long Password Denial Of Service	Improper handling of long passwords during hashing leads to memory/CPU exhaustion and DoS.	High
117	Uncontrolled Format String	Format string attacks allow attackers to read stack traces, access memory, or cause crashes.	Medium

118	Google Chrome Logger Information Disclosure	Chrome Logger HTTP header may expose sensitive server-side debug data.	Low
119	Java Virtual Machine (JVM) Version Disclosure	JVM version in server headers enables version-specific attacks.	Low
120	Missing Subresource Integrity Check	Missing SRI allows CDN compromise to inject malicious content into all dependent sites.	Info
121	HTTP Request Smuggling	Specially crafted HTTP messages can bypass security controls and firewall checks.	High
122	Possible Slowloris DOS Attack	Slow partial HTTP header requests can exhaust server resources and cause DoS.	Medium
123	Link Injection	Injected link tags enable phishing, malicious redirects, and credential stuffing.	Medium
124	Iframe Injection	Injected iframes can load third-party content enabling phishing and backdoor downloads.	Medium
125	XML External Entity DOS Attack	XXE entity expansion causes heavy server load resulting in a DoS attack.	High
126	XML External Entity (XXE) Injection	XXE injection allows confidential data disclosure, SSRF, and DoS attacks.	High
127	Oracle WebLogic URI Attack	URI path abuse in Oracle WebLogic enables RCE, XSS, and data exfiltration.	High
128	Web Cache Poisoning Attack	Cache poisoning combined with injection attacks leads to XSS, cookie stealing, and session hijack.	High
129	Microsoft Exchange Server RCE Vulnerability	SSRF flaw in Exchange servers allows authentication bypass and arbitrary code execution.	High
130	Possible BREACH Vulnerability	HTTP-level compression flaw allows MITM attackers to extract CSRF tokens and sensitive data.	Medium
131	HTTP.sys Remote Code Execution Vulnerability	Flaw in HTTP.sys parsing allows remote attackers to crash or restart the server.	High

132	GNU glibc Remote Heap Buffer Overflow (CVE-2015-0235)	GHOST vulnerability in glibc allows RCE and system compromise via vulnerable PHP.	Medium
133	Partial user controllable script source	User-controlled script src attributes enable XSS and Reverse Clickjacking attacks.	Medium
134	Incorrect Session Timeout	Sessions not terminated after inactivity allow compromised tokens to be reused.	Medium
135	JWT misconfiguration	Improperly configured JWT tokens allow account takeover attacks.	Medium
136	Permissive Client Access Policy File Detected	Misconfigured ClientAccessPolicy.xml grants unauthorized cross-domain data access.	Low
137	Weak TLS CBC cipher Detected	CBC ciphers in TLS 1.0/1.1/1.2 are vulnerable to ZOMBIE POODLE and GOLDENDOODLE attacks.	Medium
138	Possible Archive File or Compression File - Log	Compression or archive log files on the server may expose sensitive application data.	Medium
139	LFI in Apache mod-cgi	LFI in Apache 2.4.49 with mod-cgi enabled allows access to arbitrary sensitive files.	High
140	Code Injection	Invalidated user-controlled parameters interpreted by the application allow arbitrary code execution.	High
141	Cross-Site Flashing (XSF)	Unvalidated user-controlled data in Flash functions enables XSS and cross-domain attacks.	High
142	Edge Side Include Injection	ESI tag injection leads to SSRF, XSS bypass of HTTPOnly cookies, and server-side DoS.	High
143	Apache Log4j RCE Vulnerability	CVE-2021-44228 allows RCE via crafted input submitted to Log4j 2.0 through 2.14.1.	Critical
144	Apache Server ETag Header Information Disclosure	ETag headers expose inode numbers, enabling information disclosure and cache poisoning.	Medium
145	Improper Session Management	Flaws in session management lead to account hijacking and authorization bypasses.	Medium

146	Insecure Direct Object References	Unverified object references allow users to access unauthorized resources.	High
147	OS Command Injection - OOB	Out-of-band OS command injection allows arbitrary command execution on the remote server.	Critical
148	XML External Entity (XXE) Injection - OOB	Out-of-band XXE allows confidential data disclosure, SSRF, and DoS attacks.	High
149	SQL Injection OOB	Out-of-band SQL injection allows recovery and modification of database data.	Critical
150	Server-Side Request Forgery (SSRF) - OOB	Out-of-band SSRF allows attackers to make server-side HTTP requests to arbitrary domains.	Medium
151	Cross Site Scripting (XSS) OOB	Out-of-band XSS allows injection of malicious scripts executed in victim browsers.	High
152	HTTP Host Header Injection OOB	Out-of-band Host header injection leads to Cache Poisoning, XSS, and routing-based SSRF.	Medium
153	Server-side template injection OOB	Out-of-band SSTI allows arbitrary code execution by injecting template directives.	Critical
154	Spring Expression Resource Access Vulnerability (RCE)	SpEL injection in Spring Cloud Function routing allows access to local resources and RCE.	High
155	Code Injection - OOB	Out-of-band code injection allows loss of confidentiality, integrity, and availability.	Critical
156	VMware Server-side Template Injection RCE (CVE-2022-22954)	SSTI in VMware Workspace ONE Access allows network-accessible RCE without authentication.	Critical
157	Password found in server response	Clear-text password in response allows credential theft via session or XSS vulnerabilities.	Medium
158	Credential found in token	Authorization tokens containing credentials can lead to full account compromise.	Medium
159	Link Injection OOB	Out-of-band link injection enables phishing, malicious redirects, and credential stuffing.	Medium

160	Path-Relative StyleSheet Import Vulnerability	Path-relative CSS links can cause XSS and CSRF token exfiltration.	Low
161	Iframe Injection OOB	Out-of-band iframe injection enables phishing, credential stuffing, and backdoor downloads.	Medium
162	Improper Token Handling (duplicate)	Improper token handling allows authentication bypass and CSRF attacks.	Medium
163	Sensitive Information Disclosure Through URL	Sensitive data in URLs can be stolen from browser history.	Low
164	Running Service (Open Port)	An open port may lead to data loss, DoS attacks, or exploitation of running services.	Info
165	Unset/Insecure X-Permitted-Cross-Domain-Policies Header	Misconfigured cross-domain policy header allows unauthorized cross-domain data access.	Low
166	Session Resumption Enabled	TLS session resumption can lead to session stealing and replay attacks.	Info
167	DNSSEC unsigned	Unsigned DNSSEC records may allow DNS spoofing and malicious activity.	Info
168	Content Injection	Content spoofing via injection presents users with modified pages under a trusted domain.	Medium
169	JWT none algorithm	JWT none algorithm flaw allows attackers to bypass signature verification and forge tokens.	Medium
170	Weak Encoding	Weak encoding allows attackers to decode and steal sensitive information.	Medium
171	Reveals Sensitive Information (Medium)	Sensitive information in requests/responses should be encoded with proper salting techniques.	Medium
172	Reveals Sensitive Information (Info)	Sensitive information in requests/responses may expose other vulnerabilities.	Info
173	Accessible By IP Address	Servers accessible by IP address may be discovered by random scanning worms.	Low

174	No CAPTCHA on login page	Missing CAPTCHA enables automated brute force and credential stuffing attacks.	Info
175	EPMM Authentication Bypass	Weak access controls allow unauthenticated attackers to gain admin privileges.	Critical
176	WebSocket URL poisoning	Unvalidated WebSocket URLs can cause XSS, information leakage, and unauthorized access.	Medium
177	BruteForce Directory/File	Brute-forced directories and files may expose sensitive information for advanced attacks.	Medium
178	Client-side Template Injection	User-controlled data in client-side templates allows arbitrary JavaScript execution.	Critical
179	Insecure transition from HTTP to HTTPS in form post (HTTP serving HTTPS form)	HTTP pages serving HTTPS forms can be hijacked via MITM to replace or spoof the form.	Medium
180	Insecure transition from HTTPS to HTTP in form post	HTTPS pages serving HTTP forms expose submitted data to interception.	Low
181	Insecure Transport	Applications accessible via HTTP without HTTPS redirect expose login credentials and sessions.	Medium
182	Cross-Site Tracing (XST)	XST combines XSS with HTTP TRACE to steal legitimate user credentials.	Medium
183	ExtJs Arbitrary File Read	Ext JS flaw allows reading arbitrary files and initiating internal HTTP service requests.	High
184	Body Parameters Accepted in Query	GET requests with sensitive parameters expose data in browser history and proxy logs.	Medium
185	PHP CGI Argument Injection Vulnerability	PHP CGI on Windows misinterprets Best-Fit characters as PHP options, enabling RCE and source disclosure.	Critical
186	Cross-Site Request Forgery (CSRF)	CSRF forces authenticated users to execute unwanted actions on behalf of attackers.	Medium
187	Unsafe third-party link (Medium)	target='_blank' without rel='noopener' allows the linked page to access the original page's Window object.	Medium

188	Path Traversal in Apache OFBiz (Critical)	Path traversal before OFBiz 18.12.14 allows reading sensitive files and potential RCE.	Critical
189	Unsafe third-party link (Low)	target='_blank' without rel='noopener noreferrer' allows cross-site scripting and phishing.	Low
190	Path Traversal in Apache OFBiz (duplicate)	Path traversal before OFBiz 18.12.14 allows reading sensitive files and potential RCE.	Critical
191	Unsafe third-party link (Low duplicate)	target='_blank' without rel='noopener noreferrer' allows XSS and phishing attacks.	Low
192	Database Connection String Detected	Exposed database connection strings containing credentials can lead to unauthorized database access.	High
193	Web Server Content Sniffing Enabled (duplicate)	MIME type misidentification allows attackers to execute scripts in another user's session context.	Low
194	Unset/Insecure X-XSS-Protection Header Vulnerability	Missing X-XSS-Protection header increases risk of XSS, data theft, and session hijacking.	Medium
195	Possible BREACH Vulnerability (Low)	HTTP compression in HTTPS responses enables inference of sensitive data such as session tokens.	Low
196	Python Code Injection	Unsanitized user input in Python code snippets allows server-side arbitrary code execution.	High
197	Client-side desync (CSD) attack	CSD attacks desynchronize the browser from the server, enabling XSS and malicious actions.	High
198	Serialized Object in HTTP Message	Serialized objects in HTTP messages can be manipulated by injecting malicious payloads.	Medium
199	LDAP Injection	LDAP payload injection in request parameters exploits LDAP query processing vulnerabilities.	Medium
200	Unset/Insecure CSP Header Vulnerability	Missing CSP header allows XSS, Clickjacking, and data injection attacks.	Low
201	HTTPS And Mixed Content Vulnerability	HTTP resources on HTTPS pages enable MITM attacks, security degradation, and browser warnings.	Medium

202	Serialized Object in HTTP Message (duplicate)	Serialized objects in HTTP messages can be manipulated by injecting malicious payloads.	Medium
203	Insecure transition from HTTP to HTTPS in form post (duplicate)	HTTP pages serving HTTPS forms can be hijacked via MITM to replace or spoof the form.	Medium
204	Insecure transition from HTTPS to HTTP in form post (duplicate)	HTTPS pages serving HTTP forms expose submitted data to interception.	Low
205	HTTP Host Header Injection OOB (duplicate)	Unvalidated Host header values lead to Cache Poisoning, XSS, and routing-based SSRF.	Medium
206	HTTP.sys Remote Code Execution (duplicate)	HTTP.sys parsing flaw allows remote attackers to crash or restart the server.	High
207	HTTP Request Smuggling (duplicate)	Crafted HTTP messages bypass security controls and firewall checks.	High
208	HTTP TRACK Method Enabled (duplicate)	HTTP TRACK method allows client cookie capture via Cross-Site Scripting.	Low
209	HTTP OPTIONS Method Enabled (duplicate)	OPTIONS method exposes a list of supported server methods.	Low
210	HTTP Verb Tampering (duplicate)	HTTP method manipulation allows authorization bypass.	Low
211	HTTP Host Header Injection (duplicate)	Unvalidated Host header values can lead to Cache Poisoning and XSS.	Medium
212	Missing HttpOnly Flag From Cookie (duplicate)	Missing HttpOnly flag allows malicious XSS code to steal cookie data.	Low
213	HTTP Basic Authentication Enabled (duplicate)	HTTP Basic Authentication transmits credentials as cleartext.	Medium
214	HTTP PUT Method Enabled (duplicate)	HTTP PUT method allows remote upload of arbitrary files.	Low
215	HTTP Response Splitting (duplicate)	CRLF injection via response splitting enables cookie forgery, CSRF theft, and XSS.	High

216	HTTP DELETE Method Enabled (duplicate)	HTTP DELETE method allows attackers to deface websites or mount DoS attacks.	Low
217	HTTP TRACE Method Enabled (duplicate)	HTTP TRACE method allows client cookie capture via Cross-Site Scripting.	Low
218	Vulnerable Next.js CVE-2025-29927	Middleware authorization bypass via manipulated x-middleware-subrequest header in Next.js.	Critical
219	Proxy Bypass via Trusting Unvalidated Host Headers	Unvalidated Host headers allow proxy bypass, internal host disclosure, and SSRF.	High
220	Detected Vulnerable JavaScript Library Version	Outdated JavaScript libraries (e.g., jQuery, Lodash) contain known XSS or DoS vulnerabilities.	Low
221	Detected Vulnerable JavaScript Framework Version	Outdated JS frameworks (e.g., AngularJS, React, Vue.js) expose applications to sandbox bypasses and XSS.	Low
222	Server-Side Renegotiation Without RFC 5746	TLS renegotiation without RFC 5746 support allows session data injection via MITM.	High
223	Client-Side Renegotiation Without RFC 5746	Client TLS renegotiation without RFC 5746 allows prefix injection attacks.	High
224	Self-signed certificate	Self-signed certificates lack trusted CA verification, enabling MITM risks and user trust loss.	High
225	Insecure WebSocket Detector	Unencrypted ws:// WebSocket endpoints allow session hijacking and malicious payload injection.	Low
226	Server-Side TLS Renegotiation Supported	Unlimited TLS renegotiation support may lead to resource exhaustion and DoS attacks.	Medium
227	Client-Initiated TLS Renegotiation Supported	Client-initiated TLS renegotiation increases DoS risk via repeated handshake requests.	Medium
228	Host Header Injection via Unvalidated Host Leading to Redirect & Protocol Manipulation	Malicious Host header values allow protocol manipulation, phishing, and security control bypass.	Critical
229	Accept Unauthenticated Request by Default	Missing authentication checks allow unauthorized access to sensitive resources and functions.	High

230	Session Not Expired After Logout	Sessions remaining active after logout allow token reuse and session hijacking.	High
231	Sensitive Credentials Exposed in Headers	Credentials in HTTP headers can be logged by intermediaries, enabling account impersonation.	High
232	No validation on third-party or internal API responses	Unvalidated external API responses can inject malicious payloads into the application.	Medium
233	Unrestricted Access to Sensitive Business Flows	No rate limiting or access controls on business operations enable financial loss and abuse.	High
234	Missing Rate Limiting on Sensitive Business Operations	Unlimited requests on sensitive flows enable resource exhaustion and financial manipulation.	High
235	Missing Business Logic Validation on Sensitive Operations	Missing validation on business logic flows enables manipulation of critical operations.	High
236	High-Frequency Trading/Scalping Vulnerability Detected	The application is vulnerable to high-frequency trading or scalping attacks, where attackers can perform rapid successive operations to gain unfair advantages in trading, booking, or other time-sensitive business operations. This can lead to market manipulation, unfair competition, and financial losses.	High
237	Reveals Sensitive Information	The application response contains sensitive information such as keys, tokens, credentials, or internal identifiers. Exposure of such data can aid attackers in gaining unauthorized access, escalating privileges, or performing further reconnaissance. This indicates a lack of proper filtering or sanitization in API or application responses.	High
238	Reveals Sensitive Information	The application response contains sensitive information such as keys, tokens, credentials, or internal identifiers. Exposure of such data can aid attackers in gaining unauthorized access, escalating privileges, or performing further reconnaissance. This indicates a lack of proper filtering or sanitization in API or application responses.	Medium
239	Exposure of API Version	The target application exposes versioned API endpoints, legacy APIs, or development environments that may indicate poor API management practices. This can lead to information disclosure about API structure, potential access to deprecated/unsupported endpoints, and exposure of development/testing environments in production.	Low
240	mTLS Authentication Not Enforced	The target server does not enforce mutual TLS (mTLS) authentication, allowing connections without client certificates. This creates a security gap where the server cannot verify the identity of connecting clients, potentially allowing unauthorized access to sensitive APIs or services.	High
241	Weak Client Certificate Key Detected	The target server accepts connections with client certificates using weak cryptographic keys during mTLS authentication. Keys smaller than 2048 bits are considered weak and vulnerable to cryptographic attacks. This indicates that the server's certificate validation is not properly enforcing minimum key strength requirements, potentially allowing compromised authentication.	Medium

242	Self-Signed Client Certificate Accepted	The target server accepts connections with self-signed client certificates during mTLS authentication. This indicates that the server's certificate validation is not properly verifying certificate authority chains, allowing untrusted certificates to authenticate successfully. Self-signed certificates cannot be verified by trusted Certificate Authorities and may indicate weak certificate validation.	High
243	Expired Client Certificate Accepted	The target server accepts connections with expired client certificates during mTLS authentication. This indicates that the server's certificate validation is not properly checking certificate expiry dates, allowing potentially compromised or outdated certificates to authenticate successfully.	High
244	Weak Client Certificate Detection	Client certificate uses weak cryptographic algorithms (MD5, SHA1, MD2, MD4) that are vulnerable to attacks. The certificate's signature or public key algorithms are cryptographically weak and can be exploited by attackers.	Medium
245	API Version Authentication Bypass	When older API versions lack authentication/authorization controls that are present in newer versions. Attackers can bypass security by targeting older, less secure API versions (e.g., /v1/, /v2/, /api/v1/, etc.) that don't enforce proper authentication mechanisms. Impact:Unauthorized access to sensitive data through older API versions - Bypass of security controls by targeting deprecated endpoints - Data leakage through unsecured legacy APIs - Privilege escalation via version-specific vulnerabilities	High
246	Open Redirect Injection Vulnerability	Detects open redirect vulnerabilities where user-controlled input in sensitive redirect parameters (redirect, redirect_to, url, next, return, continue, dest, target, callback, etc.) is reflected in HTTP responses without proper validation, potentially allowing attackers to redirect users to malicious external domains. Attackers can redirect users to malicious websites for phishing, credential theft, malware distribution, or bypassing security controls. Can be used in conjunction with other attacks like CSRF or social engineering.	Medium
247	Host Injection in HTML Attributes	Detects host injection vulnerabilities where user-controlled input is reflected in HTML attributes within the response body, potentially allowing attackers to manipulate client-side behavior, perform cache poisoning, or bypass security controls through DOM manipulation.	Medium
248	URI Path Injection Vulnerability	Vulnerability occurs when user-supplied input is improperly handled in URI paths, allowing attackers to inject arbitrary content or perform path traversal attacks. The application reflects user-controlled data in URI paths without proper validation or sanitization.	High
249	Object Reference Injection Vulnerability	Object Reference Injection is a vulnerability that occurs when applications expose internal object references (such as database IDs, file handles, or resource identifiers) in URLs or parameters without proper authorization checks. Attackers can manipulate these references to access unauthorized resources or perform actions on objects they shouldn't have access to. This vulnerability is particularly dangerous in APIs and web applications that use predictable object identifiers.	Medium

250	Server-Side Request Forgery (SSRF) - URL Normalization Bypass	Server-Side Request Forgery (SSRF) vulnerability occurs when an application doesn't properly normalize or canonicalize user-supplied URLs, allowing attackers to bypass security filters using different representations of the same IP address. This enables attackers to make requests to internal systems, cloud metadata services, or other restricted resources.	High
251	No Rate Limiting on API Endpoints	The API endpoint does not implement rate limiting mechanisms, allowing attackers to send unlimited requests in rapid succession. After multiple consecutive requests (default: 10 attempts), the system shows no protection against rapid successive operations, making it vulnerable to resource exhaustion attacks, DDoS, and abuse.	High
252	No Pagination or Limits on Large Dataset APIs	The API endpoint returns large datasets without implementing pagination or limits, allowing a single request to retrieve unbounded records (100+ items or 50KB+ responses). The plugin detects this by identifying large dataset APIs (list, search, query endpoints) and verifying absence of pagination indicators in both request parameters and response structure.	Medium
253	Continuous OTP/Email/SMS Resend APIs Without Rate Limiting	The API endpoint allows continuous resending of OTP (One-Time Password), email verification codes, or SMS messages without implementing rate limiting or throttling mechanisms. The plugin identifies resend APIs and tests for continuous resend capability by sending multiple consecutive requests (5+ attempts) and checking for absence of rate limiting controls.	Medium
254	Adobe Commerce/Magento - Vulnerable Version Disclosure (CVE-2025-54236)	This vulnerability detection identifies Adobe Commerce and Magento Open Source installations that disclose their version information through HTTP headers or HTML content. The disclosed version indicates the application is running a vulnerable version (2.4.9-alpha2 or earlier) that is susceptible to CVE-2025-54236 (SessionReaper). CVE-2025-54236 is a critical vulnerability in Adobe Commerce and Magento Open Source that allows unauthenticated attackers to hijack customer sessions through improper input validation in the Commerce REST API's ServiceInputProcessor component.	Critical
255	Adobe Commerce/Magento - Improper Input Validation in ServiceInputProcessor (CVE-2025-54236)	This vulnerability detection actively probes Adobe Commerce and Magento Open Source REST API endpoints to confirm the presence of CVE-2025-54236 (SessionReaper) vulnerability. The plugin sends non-destructive serialized PHP markers to REST API endpoints and analyzes responses for ServiceInputProcessor error patterns that indicate the vulnerability is present. CVE-2025-54236 is a critical vulnerability in Adobe Commerce and Magento Open Source that allows unauthenticated attackers to hijack customer sessions through improper input validation in the Commerce REST API's ServiceInputProcessor component. The vulnerability exists in versions 2.4.9-alpha2 and earlier.	Critical
256	Excessive CPU/Memory Load on API Endpoints	The API endpoint performs CPU/memory intensive operations (such as image generation, PDF rendering, video transcoding, encryption, etc.) and responds slowly without apparent resource limits. This allows attackers to exhaust server resources through repeated requests, leading to denial of service conditions.	High
257	Missing Timeout Controls on API Endpoints	The API endpoint lacks timeout controls, allowing requests to run for extended periods (30+ seconds) without being terminated. This enables resource exhaustion attacks where attackers can send requests that consume server resources indefinitely or for very long duration.	High

258	API Accepts Deeply Nested JSON/XML Structures	The API endpoint accepts and processes deeply nested JSON or XML structures (50+ levels of nesting) without limits. This can lead to parser exhaustion, stack overflow errors, and denial of service attacks. Deeply nested structures can cause: - Parser stack overflow - Excessive memory consumption - CPU exhaustion during parsing - Application crashes - Denial of service conditions	High
259	LLM Leakes the system prompt or Insturctions	This vulnerability occurs when an AI system unintentionally exposes its system prompt. It can contain hidden instructions, rules, or roles that control the modelâ€™s behaviour. If leaked, the attacker may gain access to internal logic, security bypass methods, or even sensitive information such as API keys, credentials, connection strings, or user-role rules mistakenly embedded in the prompt. These information can be used by attacker differently or for the prompt injection purpose to alter the behaviour of the LLM.	High
260	LLM Sensitive Data Exposure Check	This vulnerability occurs when an AI system unintentionally exposes sensitive information such as API keys, authentication tokens, credentials, connection strings, or other secrets in its responses. This can happen when LLMs access backend systems, databases, or configuration files through function calls, tool usage, or when sensitive data is mistakenly embedded in training data or system prompts. Attackers can exploit prompt injection techniques to trick the LLM into revealing these secrets, which can then be used for unauthorized access, data breaches, or further exploitation of the system.	High
261	Server-Side Request Forgery (SSRF) via Unusual URI Scheme	This vulnerability detects when an application is vulnerable to Server-Side Request Forgery (SSRF) attacks through the acceptance of unusual URI schemes. Many applications implement SSRF protection by only allowing <http://> and <https://> schemes, but some applications may accept other schemes like gopher://, <ftp://,> dict://, ldap://, ws://, or wss://, which can be used to bypass SSRF filters or access internal services.	Medium

262	GCP Metadata Service Accessible via SSRF	This vulnerability detects when an application is vulnerable to Server-Side Request Forgery (SSRF) attacks that allow access to the Google Cloud Platform (GCP) Compute Engine Instance Metadata Service. The GCP metadata service is accessible at metadata.google.internal or 169.254.169.254 and provides sensitive information about the compute instance, including project details, service account tokens, and instance configuration.	Critical
263	Azure Metadata Service Accessible via SSRF	This vulnerability detects when an application is vulnerable to Server-Side Request Forgery (SSRF) attacks that allow access to the Microsoft Azure Instance Metadata Service (IMDS). The Azure metadata service is accessible at the link-local IP address 169.254.169.254 and provides sensitive information about the Azure VM, including managed identity tokens, subscription details, and VM configuration.	Critical
264	Server-Side Request Forgery (SSRF) via HTTP Header Injection	This vulnerability detects Server-Side Request Forgery (SSRF) attacks that occur when HTTP headers are manipulated to force the server to make requests to internal systems or cloud metadata services. Unlike URL parameter-based SSRF, this attack vector exploits header injection vulnerabilities.	High
265	Open Redirection Chained with SSRF	This vulnerability detects when an application has an open redirect vulnerability that can be chained with Server-Side Request Forgery (SSRF) attacks to bypass SSRF protections and access internal systems. This is a critical vulnerability because open redirects can be used as a bypass mechanism to circumvent SSRF protections that validate URLs before making requests.	Critical
266	React2Shell (CVE-2025-55182) - Active Detection: RCE Vulnerability Confirmed	CVE-2025-55182, also known as "React2Shell," is a critical remote code execution (RCE) vulnerability in React Server Components (RSC) that allows unauthenticated attackers to execute arbitrary code on affected servers. This active detection method confirms the vulnerability by sending specially crafted payloads and analyzing responses for exploitation indicators.	Critical

267	React2Shell (CVE-2025-55182) - Passive Detection: Vulnerable Version Disclosure	CVE-2025-55182, also known as "React2Shell," is a critical remote code execution (RCE) vulnerability in React Server Components (RSC) that allows unauthenticated attackers to execute arbitrary code on affected servers. This passive detection method identifies vulnerable applications by detecting version disclosure in HTTP headers and HTML responses without sending any attack payloads.	Critical
268	Apache Tika XXE (CVE-2025-66516) - Active Detection: XXE Vulnerability Confirmed	CVE-2025-66516 is a critical XML External Entity (XXE) injection vulnerability in Apache Tika that allows attackers to read local files, trigger out-of-band requests, and cause Denial of Service (DoS) attacks through crafted XFA (XML Forms Architecture) files embedded in PDF documents. This active detection method confirms the vulnerability by uploading malicious PDF files and analyzing responses for exploitation indicators.	Critical
269	Apache Tika XXE (CVE-2025-66516) - Passive Detection: Vulnerable Version Disclosed	CVE-2025-66516 is a critical XML External Entity (XXE) injection vulnerability in Apache Tika that affects versions 1.13 through 3.2.1. This passive detection method identifies vulnerable versions by analyzing version information disclosed in HTTP responses, without performing any exploitation attempts.	High
270	Stack Trace Exposure in HTTP Response	When an application encounters an error or exception, it may return detailed stack trace information in the HTTP response body. This information is intended for developers during debugging but should be suppressed in production environments. Attackers can leverage exposed stack traces to: Understand Application Architecture: Stack traces reveal class names, package structures, and method hierarchies Identify Vulnerable Components: File paths and line numbers can indicate outdated libraries or vulnerable code sections Map Internal Structure: Directory structures and file naming conventions are exposed Plan Targeted Attacks: Knowledge of framework versions and internal logic aids in crafting specific exploits	Medium

271	React2Shell (CVE-2025-55182) - OOB	CVE-2025-55182, also known as "React2Shell," is a critical remote code execution (RCE) vulnerability in React Server Components (RSC) that allows unauthenticated attackers to execute arbitrary code on affected servers. This active detection method confirms the vulnerability by sending specially crafted payloads and analyzing responses for exploitation indicators.	Critical
272	RSC - Passive Detection: DOS Vulnerability	CVE-2025-55184, CVE-2025-67779 and CVE-2026-23864 are high-severity denial of service (DoS) vulnerabilities in React Server Components (RSC) that allow malicious HTTP requests to cause infinite loops, hanging the server process and consuming CPU resources. This passive detection method identifies vulnerable applications by detecting version disclosure in HTTP headers and HTML responses without sending any attack payloads.	High
273	RSC Source Code Exposure (CVE-2025-55183)	CVE-2025-55183 is a medium-severity source code exposure vulnerability in React Server Components (RSC) that allows malicious HTTP requests to unsafely return the source code of Server Functions, potentially exposing secrets hardcoded in source code. This passive detection method identifies vulnerable applications by detecting version disclosure in HTTP headers and HTML responses without sending any attack payloads.	Medium
274	Predictable UUIDs or tokens	This vulnerability detects predictable UUIDs or token-style identifiers returned by API endpoints. If an application uses low-entropy, patterned, or otherwise predictable UUIDs/tokens as object references, attackers can guess/enumerate valid identifiers and access objects they should not be authorized to access (BOLA/IDOR-style impact).	Medium
275	Missing Authorization for UI-Exposed Sensitive API	This vulnerability detects sensitive APIs that are exposed to the UI (frontend) and return sensitive data without any authorization headers. When UI-facing APIs serving user data, admin operations, or financial information are callable directly from the browser without tokens, cookies, or API keys, attackers can invoke them from their own browser or scripts to access data or perform actions theyâ€™re not entitled to.	High

276	Session ID in Persistent Cookies	Session ID cookies are set as persistent cookies (with Expires or Max-Age attributes). Persistent cookies are stored on disk and survive browser restarts increasing the risk of session hijacking. Impact: Persistent session cookies can be stolen from disk storage and reused by attackers even after the browser is closed. This extends the attack window for session hijacking attacks and increases the risk of unauthorized access.	Medium
277	Type Confusion Vulnerability	This vulnerability detects when an API accepts incorrect data types, leading to type confusion errors, logic flaws, or unexpected behavior. Type confusion occurs when an API expects one data type (e.g., integer) but accepts another (e.g., string), potentially causing application crashes, logic errors, or security vulnerabilities.	Medium
278	Invalid/Missing Keys Vulnerability	This vulnerability detects when an API fails to properly handle invalid or missing keys in request payloads. APIs that don't validate required fields or reject invalid key names can lead to application errors, logic flaws, or security vulnerabilities. This includes missing required fields, invalid key names with special characters, and duplicate key handling issues.	Medium
279	Enum Validation Vulnerability	This vulnerability detects when an API fails to validate enumerated fields (status, type, state, etc.), allowing invalid enum values to be accepted. APIs that don't properly validate enum-like fields can lead to logic flaws, unexpected behavior, or security vulnerabilities. This includes accepting invalid enum values, no validation of enum-like fields, and logic flaws from invalid enum acceptance.	Medium
280	Webhook Validation Vulnerability	This vulnerability detects when webhook endpoints accept requests without proper signature validation (e.g., missing HMAC). Webhooks are HTTP callbacks that allow external services to notify an application of events. Without proper signature validation, attackers can forge webhook requests, potentially triggering unauthorized actions, data manipulation, or service abuse.	High

281	Long Redirection Response	This vulnerability detects when HTTP redirect responses (3xx status codes) contain unusually long response bodies. Normal redirect responses should be minimal (typically less than 500 bytes) as they only need to provide the Location header. Long redirect responses can indicate: Information leakage in redirect responses Potential DoS vectors through large redirect payloads Improper redirect handling that exposes sensitive data Security misconfigurations in redirect mechanisms	Low
282	Content Type Incorrectly Stated	This vulnerability detects when the Content-Type header doesn't match the actual response content. Mismatched content types can lead to: MIME type confusion attacks where browsers interpret content differently than intended XSS vulnerabilities when HTML/JavaScript is served with incorrect content types Content sniffing issues where browsers override declared content types Security bypasses through incorrect MIME type handling	Low
283	Token Leakage via Referer	The HTTP Referer header is sent automatically by the browser when the user follows a link or when a page loads cross-origin resources (e.g. scripts, images, analytics). If the referring URL contained a sensitive token (in query string or path), the browser sends that full URLâ€”including the tokenâ€”to the destination server. When the destination is a third party (e.g. analytics, ads, or an external site), the token is disclosed to an unauthorized actor. That is information disclosure (CWE-200) and misuse of GET/URL for sensitive data (CWE-598).	Low

284	Self-Redirection	This vulnerability detects when an application redirects to itself (same URL), which can indicate**: - Infinite redirect loops that can cause DoS 1. Logic errors in redirect handling 2. Potential for redirect-based attacks 3. Security misconfigurations in redirect mechanisms	Low
285	Options Bleed Apache (CVE-2017-15710)	This vulnerability detects the Apache Options Bleed vulnerability (CVE-2017-15710) which allows information disclosure through the HTTP OPTIONS method response. The vulnerability occurs when Apache servers leak sensitive information in OPTIONS method responses, including**: - File system paths Server configuration details Internal application structure Sensitive data in response bodies	Low
286	Web Service Disclosed	This vulnerability detects when web services (WSDL, SOAP, REST API documentation) are exposed without proper access controls, revealing**: - API endpoints and methods Request/response schemas Service implementation details Internal API structure Authentication mechanisms	Low
287	Deprecated API Behavior Inconsistency	This vulnerability detects when deprecated APIs respond differently than documented ones. Deprecated API versions may have different behavior, error handling, or response formats compared to current versions, which can lead to security issues, unexpected behavior, or confusion for API consumers.	Medium

288	Authorization Before Parsing or Validation	This vulnerability detects when APIs check authentication but process invalid or malformed input. This indicates that authorization happens before input parsing/validation, which is a security vulnerability. Malformed input should be rejected before authentication checks to prevent resource consumption and potential exploitation.	High
289	Serialized Object in HTTP Message OOB	This vulnerability detects insecure deserialization by sending crafted serialized objects (e.g., Apache Shiro rememberMe cookie) that trigger out-of-band requests to a collaborator server upon deserialization. This is particularly effective for detecting blind deserialization vulnerabilities where the application deserializes untrusted data without proper validation, such as CVE-2016-4437 (Apache Shiro deserialization).	Critical
290	Insecure Direct Object References (IDOR)	Insecure Direct Object References (IDOR) occur when an application provides direct access to objects based on user-supplied input without proper authorization checks. Attackers can manipulate object identifiers (user IDs, resource IDs, etc.) to access unauthorized resources, leading to data exposure, privilege escalation, and unauthorized data modification.	High
291	Deprecated or Beta APIs Still Exposed	This vulnerability detects when deprecated or beta API versions are still exposed and accessible. Deprecated or beta APIs should be removed or restricted, as they may contain known vulnerabilities or lack security patches present in current versions. When these APIs remain accessible, attackers can exploit unpatched vulnerabilities or use outdated endpoints with weaker security controls.	Medium
292	Missing Custom Error Page (CWE-756)	This vulnerability detects when 404, 500, or 403 error pages expose server version information, technology stack details, or other sensitive information instead of using custom error pages. Default error pages often reveal technology stack, server versions, and internal paths that should be hidden from attackers.	Medium

293	Default Error Page from Web Server or Framework	This vulnerability detects when default error pages from web servers (Nginx, Apache, Tomcat, etc.) or frameworks (Django, Flask, Rails, Laravel, etc.) are returned instead of custom error pages. Default error pages often reveal server software, versions, and configuration details that should be hidden from attackers.	Medium
294	Body Parameters Accepted in Query	When an application accepts POST requests with body parameters but also processes the same parameters when sent as GET (in the query string), it is said to accept â€œbody parameters in queryâ€ or to allow an insecure transition from POST to GET. The server is effectively treating GET and POST the same for that endpoint, which violates the usual expectation that sensitive or state-changing data is sent in the request body and not in the URL. This behavior leads to: CSRF via GET â€“ A single link or image load can trigger the action; no form POST or JavaScript required. Caching and logging of sensitive parameters â€“ Proxies, CDNs, and servers often log or cache URLs; parameters in the query string are exposed. Referrer leakage â€“ When the user follows a link to another site, the full URL (including query parameters) can be sent in the Referer header. Exposure in browser history and bookmarks â€“ Sensitive tokens or identifiers end up in history and shared links.	High
295	Cross-Site Request Forgery (CSRF)	Cross-Site Request Forgery (CSRF) allows an attacker to trick a victimâ€™s browser into sending unwanted requests to a target site using the victimâ€™s session. Because the browser automatically sends cookies and other credentials with same-site requests, the server cannot distinguish a legitimate user action from a request triggered by a malicious pageâ€”unless it validates Origin/Referer or requires a CSRF token. The POC describes how the scanner detects CSRF using DAST logic: re-sending the same request with the crawlerâ€™s session (cookies/auth headers) but with Referer and Origin set to a malicious origin. If the server returns the same success response, it is not validating origin and is vulnerable to CSRF.	High

296	Cross-Site Request Forgery (CSRF)	Cross-Site Request Forgery (CSRF) allows an attacker to trick a victimâ€™s browser into sending unwanted requests to a target site using the victimâ€™s session. Because the browser automatically sends cookies and other credentials with same-site requests, the server cannot distinguish a legitimate user action from a request triggered by a malicious pageâ€”unless it validates Origin/Referer or requires a CSRF token. The POC describes how the scanner detects CSRF using DAST logic: re-sending the same request with the crawlerâ€™s session (cookies/auth headers) but with Referer and Origin set to a malicious origin. If the server returns the same success response, it is not validating origin and is vulnerable to CSRF.	Low
297	Flash Cross-Domain Policy Found	A Flash cross-domain policy (also called Adobe cross-domain policy) is an XML file, usually named crossdomain.xml and placed at the site root or other well-known path. It was introduced by Macromedia/Adobe for Flash Player and defines which domains are allowed to: Load SWF content from the application. Make cross-domain HTTP requests to the application (e.g. from Flash or legacy clients).	Low
298	Server-Side Code Injection DoS (ReDoS / eval-based)	Server-Side Code Injection DoS is a vulnerability where the application executes user-supplied input as server-side code. Typically, a request parameter (often in a JSON body) is passed into a code-execution context such as eval, Function, or a JavaScript sandbox. Because the input is interpreted as code, an attacker can supply input that causes the server to consume excessive CPU (e.g. ReDoS), throw an unhandled exception, or hangâ€”resulting in Denial of Service.	High
299	Missing DNS CAA Record (RFC 8659)	DNS Certification Authority Authorization (CAA) is a DNS record type defined in RFC 8659 that lets a domain owner specify which Certificate Authorities (CAs) are permitted to issue TLS certificates for that name. When CAA is absentâ€”or when records exist but do not include an issue or issuewild tagâ€”any publicly-trusted CA may issue certificates for the domain. That widens the trust surface for certificate mis-issuance, rogue-CA abuse, and BGP/DNS hijack-driven domain-validation attacks.	Low

300	Lack of Per-IP Rate Limiting on API	Authentication endpoints (login, OTP, password reset, token refresh) and abuse-prone APIs are common targets for credential stuffing and automated abuse. Rate limiting keyed only to username or API key can be circumvented by changing the perceived client IP on each request. Behind reverse proxies, applications often read X-Forwarded-For or similar headers. If throttling uses those values without validating the proxy chain, an attacker can send many requests that appear to originate from different IPs.	Medium
301	Lack of Per-Device / User-Agent Rate Limiting on API	The alert Lack of Per-Device / User-Agent Rate Limiting on API is reported when an API endpoint accepts repeated successful traffic while the User-Agent changes on every request and no rate-limit response is observed. Per-device / User-Agent rate limiting is the expected control: the application tracks how many requests a given client fingerprint (typically derived from User-Agent and optional device headers) may perform in a time window, and blocks or slows further calls from that fingerprint. Lack of that control means rotating the UA is enough to evade the throttle bucket.	Medium
302	Privilege Escalation via User-Submitted Body Flags	The alert Privilege Escalation via User-Submitted Body Flags (CWE-807) is reported when elevating a privilege-related field already in the request body produces a 2xx response that materially differs from the original and introduces new privilege-related content. Expected control: roles and entitlements are assigned server-side from identity stores or admin workflows; clients may send display fields but must not control authorization. Failure: the API honors tampered role, isAdmin, isVIP, or similar keys in the request body.	High
303	Privilege Escalation via Trusted Client Headers	The alert Privilege Escalation via Trusted Client Headers (CWE-807) is reported when injecting an elevated custom header produces a 2xx response that materially differs from the baseline and introduces new privilege-related content. Expected control: identity and tenant context come from validated tokens or server-side session state; internal headers are set only by a trusted edge and removed before user-facing routes. Failure: the API treats	High

		client-supplied X-Role, X-User-Type, X-Org-Id, or similar headers as truth.	
304	Negative or Zero Pricing Accepted on Purchase Flow	The alert Negative or Zero Pricing Accepted on Purchase Flow (CWE-840) is reported when tampering a positive pricing field in the request body to a negative, zero, or micro-cent value still yields a 2xx response with transaction-success markers and without validation-rejection or decline text. Expected control: the server ignores client price fields for authorization of charge amount and derives totals from trusted catalog/cart state. Failure: the API persists or processes the tampered amount as the order total.	High
305	Duplicate Discount / Coupon Redemption Allowed	The alert Duplicate Discount / Coupon Redemption Allowed (CWE-840) is reported when the scanner sends the same checkout body three times and each response is 2xx with transaction-success markers, with no duplicate-rejection phrasing on any replay. Expected control: each code is redeemed once per policy (globally or per account), enforced in the database or promotion service before order finalization. Failure: the API completes multiple orders with the same coupon field unchanged.	High