Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Injection Vulnerabilities				
1	SQL Injection (SQLi)	A type of security vulnerability that allows attackers to manipulate SQL queries executed by a web application's database.	Inadequate input validation and improper use of SQL queries without parameterization or prepared statements.	Unauthorized access to sensitive data, data manipulation, database corruption, or complete system compromise.	Use parameterized queries or prepared statements to prevent SQL Injection. Implement input validation and sanitize user input to filter out malicious SQL code.
2	Cross-Site Scripting (XSS)	A vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users.	Lack of proper input validation and output encoding in web applications.	Session hijacking, phishing attacks, cookie theft, defacement of web pages, or malware distribution.	Implement strict input validation, sanitize user input, and use output encoding to prevent XSS attacks. Use Content Security Policy (CSP) headers to restrict the execution of scripts.
3	Cross-Site Request Forgery (CSRF)	A vulnerability that allows attackers to trick authenticated users into executing unintended actions on web applications.	Lack of CSRF tokens or inadequate validation of requests.	Unauthorized actions performed by authenticated users without their consent, such as fund transfers or account deletions.	Implement CSRF tokens, validate and compare request tokens, use anti-CSRF tokens in forms, and employ same-site cookie attributes to mitigate CSRF attacks.
4	Remote Code Execution (RCE)	A vulnerability that allows attackers to execute arbitrary code on a target system remotely.	Insecure input validation, lack of proper access controls, and vulnerabilities in software components or services.	Complete compromise of the target system, unauthorized data access, data manipulation, or installation of malware.	Implement strict input validation, perform regular security updates and patches, apply the principle of least privilege, and employ network segmentation to mitigate RCE attacks.
5	Command Injection	A vulnerability that allows attackers to execute arbitrary system commands on a target system.	Inadequate input validation and improper handling of user-controlled input in system commands.	Unauthorized access to system resources, data leakage, system compromise, or execution of arbitrary commands.	Implement strict input validation, use parameterized queries or prepared statements, sanitize user input, and avoid executing system commands with user-controlled input to mitigate Command Injection.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
6	XML Injection	A vulnerability that allows attackers to manipulate XML input to exploit weaknesses in XML processing functionality.	Lack of proper input validation and inadequate XML parsing techniques in applications.	Information disclosure, data manipulation, denial of service, or execution of arbitrary code.	Implement strict input validation, use secure XML parsing libraries, sanitize user input, and apply appropriate access controls to mitigate XML Injection attacks.
7	LDAP Injection	A vulnerability that allows attackers to manipulate LDAP (Lightweight Directory Access Protocol) queries used for authentication and authorization.	Inadequate input validation and improper use of unfiltered user input in LDAP queries.	Unauthorized access to sensitive data, user account compromise, or denial of service.	Implement strict input validation, use parameterized LDAP queries, and sanitize user input to prevent LDAP Injection.
8	XPath Injection	A vulnerability that allows attackers to manipulate XPath queries used for XML data selection and manipulation.	Lack of proper input validation and inadequate handling of user-controlled input in XPath queries.	Unauthorized access to sensitive data, XML data manipulation, or denial of service.	Implement strict input validation, use parameterized XPath queries, and sanitize user input to prevent XPath Injection.
9	HTML Injection	A vulnerability that allows attackers to inject malicious HTML code into web pages viewed by other users.	Inadequate input validation and improper output encoding in web applications.	Defacement of web pages, phishing attacks, session hijacking, or the execution of malicious scripts.	Implement strict input validation, sanitize user input, and use output encoding to prevent HTML Injection.
10	Server-Side Includes (SSI) Injection	A vulnerability that allows attackers to execute malicious code or scripts on a web server by injecting commands into Server-Side Include (SSI) statements.	Inadequate input validation and improper use of unfiltered user input in SSI statements.	Unauthorized access to sensitive data, server compromise, or the execution of arbitrary code.	Implement strict input validation, use secure SSI configuration settings, and avoid using user-controlled input in SSI statements to prevent SSI Injection.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
11	OS Command Injection	A vulnerability that allows attackers to execute arbitrary operating system commands on a target system.	Lack of input validation and improper handling of user-controlled input in operating system commands.	Unauthorized access to system resources, data leakage, system compromise, or execution of arbitrary commands.	Implement strict input validation, use parameterized system commands, sanitize user input, and avoid executing system commands with user-controlled input to mitigate OS Command Injection.
12	Blind SQL Injection	A variant of SQL Injection where attackers can infer information from the database without directly viewing it.	Inadequate input validation and improper use of unfiltered user input in SQL queries.	Information leakage, data manipulation, or unauthorized access to sensitive data.	Implement time-based or boolean-based SQL Injection detection techniques, perform regular security audits, and use parameterized queries to prevent Blind SQL Injection.
13	Server-Side Template Injection (SSTI)	A vulnerability that allows attackers to inject malicious code or templates into server-side template engines used for dynamic content generation.	Lack of input validation and improper handling of user-controlled input in server-side template engines.	Server compromise, data leakage, or execution of arbitrary code on the server.	Implement strict input validation, use secure template engine configurations, and avoid using user-controlled input in template files to prevent Server-Side Template Injection.
	Broken Authentication and Session Management				
14	Session Fixation	A vulnerability that allows an attacker to hijack a user's session by fixing or manipulating the session ID before the user logs in.	Insecure session management and improper handling of session identifiers.	Unauthorized access to user accounts, session hijacking, data theft, or privilege escalation.	Implement secure session management practices such as using session tokens that are regenerated upon login, enforcing HTTPS, and validating session IDs to prevent Session Fixation attacks.
15	Brute Force Attack	A cyberattack method that involves systematically trying all possible combinations of passwords or keys until the correct one is found.	Weak password policies, lack of account lockout mechanisms, or inadequate protection against automated login attempts.	Account compromise, data theft, unauthorized access, or denial of service.	Implement account lockout mechanisms, enforce strong password policies, use multi-factor authentication, and monitor for suspicious login attempts to mitigate Brute Force Attacks.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
16	Session Hijacking	A cyberattack where an attacker gains unauthorized access to a user's session by stealing or guessing their session ID.	Insecure transmission of session tokens, session fixation vulnerabilities, or weak session management practices.	Unauthorized access to user accounts, data theft, privilege escalation, or impersonation.	Implement secure session management, use HTTPS, enforce the use of secure cookies, and regularly rotate session IDs to prevent Session Hijacking attacks.
17	Password Cracking	A technique used to uncover passwords by systematically trying all possible combinations until the correct one is found.	Weak or predictable passwords, inadequate password policies, or insufficient encryption techniques to protect stored passwords.	Account compromise, data theft, unauthorized access, or privilege escalation.	Implement strong password policies, use password hashing algorithms with salting, enforce regular password changes, and educate users about creating strong and unique passwords to prevent Password Cracking attacks.
18	Weak Password Storage	A vulnerability that arises when passwords are stored in an insecure manner, such as plaintext or with weak encryption methods.	Lack of encryption, inadequate hashing algorithms, or improper storage practices.	Account compromise, data breach, unauthorized access, or identity theft.	Implement strong encryption methods such as bcrypt or Argon2 for password hashing, enforce the use of salt, and securely store passwords in hashed form to prevent Weak Password Storage vulnerabilities.
19	Insecure Authentication	A vulnerability that occurs when authentication mechanisms are not properly implemented or enforced.	Lack of multi-factor authentication, weak authentication protocols, or improper session management.	Account compromise, unauthorized access, data breaches, or identity theft.	Implement multi-factor authentication, use strong encryption for authentication data, enforce secure authentication protocols (such as OAuth), and regularly audit authentication mechanisms to prevent Insecure Authentication vulnerabilities.
20	Cookie Theft	A cyberattack where an attacker steals a user's cookies, often through techniques like session hijacking or XSS attacks.	Insecure transmission of cookies, session fixation vulnerabilities, or XSS vulnerabilities in web applications.	Unauthorized access to user accounts, session hijacking, data theft, or impersonation.	Implement secure cookie handling practices, use HTTPS, enforce HttpOnly and Secure flags for cookies, and regularly rotate session IDs to prevent Cookie Theft attacks.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
21	Credential Reuse	A vulnerability that arises when users reuse the same username and password across multiple accounts or services.	Poor password management practices, lack of awareness about the risks of credential reuse, or inadequate security education.	Account compromise, data breaches, identity theft, or unauthorized access to multiple accounts.	Educate users about the risks of credential reuse, enforce strong password policies, encourage the use of password managers, and implement multi-factor authentication to mitigate Credential Reuse vulnerabilities.
	Sensitive Data Exposure				
22	Inadequate Encryption	A vulnerability that occurs when sensitive data is not adequately encrypted, making it vulnerable to unauthorized access or theft.	Lack of encryption protocols, weak encryption algorithms, or improper key management.	Data breaches, unauthorized access to sensitive information, or exposure of confidential data.	Implement strong encryption algorithms (e.g., AES), use secure encryption protocols (e.g., TLS/SSL), enforce proper key management practices, and regularly audit encryption mechanisms to ensure adequate protection of sensitive data against Inadequate Encryption vulnerabilities.
23	Insecure Direct Object References (IDOR)	A vulnerability where an attacker can access and manipulate unauthorized data objects directly via object references.	Improper access control mechanisms, lack of validation or authorization checks on object references, or predictable object identifiers.	Unauthorized access to sensitive data, exposure of confidential information, or compromise of user privacy.	Implement proper access controls, perform input validation and authorization checks, use unique and unpredictable identifiers for objects, enforce the principle of least privilege, and conduct thorough security assessments to identify and mitigate Insecure Direct Object References (IDOR) vulnerabilities.
24	Data Leakage	A vulnerability that occurs when sensitive data is inadvertently disclosed or exposed to unauthorized parties.	Inadequate data protection measures, weak access controls, or improper handling of sensitive information.	Breach of privacy, loss of sensitive information, financial damage, reputation loss, or regulatory penalties.	Implement robust access controls, use encryption for sensitive data, employ data loss prevention (DLP) solutions, regularly audit data handling processes, and provide security awareness training to prevent Data Leakage vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
25	Unencrypted Data Storage	A vulnerability that arises when data is stored without encryption, making it susceptible to unauthorized access or theft.	Failure to implement encryption mechanisms, lack of awareness about data security best practices, or inadequate security controls.	Unauthorized access to sensitive data, data breaches, exposure of confidential information, or regulatory non-compliance.	Implement encryption for data at rest (e.g., using full-disk encryption, database encryption), enforce encryption policies, use secure key management practices, regularly assess data storage systems, and educate personnel on the importance of data encryption to mitigate Unencrypted Data Storage vulnerabilities.
26	Missing Security Headers	A vulnerability where critical security headers, such as Content-Security-Policy (CSP) or X-Frame-Options, are absent from web responses.	Inadequate configuration of web servers, lack of awareness about security headers, or failure to implement security best practices.	Increased risk of XSS attacks, clickjacking, data breaches, or unauthorized access to sensitive information.	Implement appropriate security headers in web server configurations, use automated tools to scan for missing headers, configure web application firewalls (WAFs) to enforce security policies, and regularly audit web applications to identify and address Missing Security Headers vulnerabilities.
27	Insecure File Handling	A vulnerability that occurs when applications handle files in an insecure manner, allowing attackers to upload or execute malicious files.	Lack of input validation, insufficient file permissions, or improper file processing mechanisms.	Unauthorized file uploads, execution of malicious code, data breaches, or compromise of system integrity.	Implement strict file upload validation, enforce file type restrictions, use secure file permissions, employ sandboxing or isolation techniques for file execution, and regularly update and patch file handling libraries to mitigate Insecure File Handling vulnerabilities.
	Security Misconfiguration				
28	Default Passwords	A vulnerability where default or weak passwords are used for authentication, making systems susceptible to unauthorized access.	Failure to change default passwords, lack of password complexity requirements, or inadequate password management practices.	Unauthorized access, data breaches, privilege escalation, or compromise of sensitive information.	Implement strong password policies, enforce password complexity requirements, use password managers, regularly rotate passwords, educate users on password security best practices, and conduct regular audits to identify and change default or weak passwords.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
29	Directory Listing	A vulnerability that occurs when a web server exposes the contents of directories, allowing unauthorized users to browse files.	Improper server configurations, lack of directory access restrictions, or misconfigured directory indexing settings.	Exposure of sensitive files, disclosure of confidential information, or unauthorized access to directory contents.	Disable directory indexing, configure proper directory access permissions, use default index files, implement secure server configurations, and regularly scan web servers for directory listing vulnerabilities to prevent unauthorized access to directory contents.
30	Unprotected API Endpoints	A vulnerability where APIs lack proper authentication or access controls, enabling attackers to interact with them without authorization.	Absence of authentication mechanisms, inadequate access controls, or poorly implemented API security measures.	Unauthorized access to sensitive data, data breaches, leakage of confidential information, or manipulation of API functionality.	Implement authentication mechanisms (e.g., OAuth, API keys), enforce access controls, use encryption for API communications, implement rate limiting and throttling, regularly audit API endpoints, and employ security testing techniques (e.g., penetration testing) to identify and remediate Unprotected API Endpoints vulnerabilities.
31	Open Ports and Services	A vulnerability where unnecessary ports and services are left open on systems, increasing the attack surface and risk of exploitation.	Failure to close unused ports, lack of port filtering or firewall rules, or improper network configurations.	Increased exposure to network attacks, unauthorized access to systems, or exploitation of vulnerabilities in open services.	Conduct port scanning and vulnerability assessments, close unnecessary ports and services, implement firewall rules and network segmentation, use intrusion detection and prevention systems (IDPS), monitor network traffic, and regularly update and patch systems to mitigate the risks associated with Open Ports and Services vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
32	Improper Access Controls	A vulnerability arising from inadequate access controls, allowing unauthorized users to gain access to sensitive resources or data.	Lack of proper authentication mechanisms, insufficient authorization checks, or misconfigured access control lists (ACLs).	Unauthorized access to sensitive data, data breaches, privilege escalation, or compromise of system integrity.	Implement strong authentication mechanisms, enforce least privilege access, use role-based access control (RBAC), regularly review and update access control policies, conduct access control audits, and employ security testing techniques to identify and remediate Improper Access Controls vulnerabilities.
33	Information Disclosure	A vulnerability that occurs when sensitive information is inadvertently disclosed or exposed to unauthorized users.	Inadequate data protection measures, weak access controls, or improper handling of sensitive information.	Breach of privacy, loss of sensitive information, financial damage, reputation loss, or regulatory penalties.	Implement data encryption, enforce strict access controls, use data anonymization techniques, conduct regular data audits, provide security awareness training, and establish incident response procedures to prevent and mitigate Information Disclosure vulnerabilities.
34	Unpatched Software	A vulnerability resulting from failure to apply security patches or updates, leaving systems vulnerable to known exploits.	Neglecting patch management processes, lack of awareness about available patches, or operational constraints preventing patching.	Exploitation of known vulnerabilities, malware infections, data breaches, or compromise of system integrity.	Implement a robust patch management program, prioritize critical security patches, automate patch deployment where possible, conduct vulnerability assessments and patch compliance checks, monitor vendor advisories, and establish procedures to quickly remediate Unpatched Software vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
35	Misconfigured CORS	A vulnerability where Cross-Origin Resource Sharing (CORS) policies are improperly configured, allowing unauthorized access to resources.	Incorrect CORS policy settings, lack of understanding of CORS security implications, or misconfigured server settings.	Unauthorized access to sensitive data, cross-site request forgery (CSRF) attacks, or data leakage between different origins.	Configure CORS policies to restrict access to trusted domains, use appropriate CORS headers (e.g., Access-Control-Allow-Origin), enforce same-origin policy, regularly audit CORS configurations, and conduct security testing to identify and address Misconfigured CORS vulnerabilities.
36	HTTP Security Headers Misconfiguration	A vulnerability that occurs when security headers in HTTP responses are not properly configured to protect web applications from various attacks.	Incorrect configuration of security headers, lack of awareness about security best practices, or failure to implement recommended security controls.	Increased susceptibility to attacks such as XSS, clickjacking, or data injection, compromising the security and integrity of web applications.	Configure security headers like Content Security Policy (CSP), X-Content-Type-Options, X-Frame-Options, X-XSS-Protection, HTTP Strict Transport Security (HSTS), and others as appropriate, regularly audit HTTP headers for misconfigurations, and utilize automated tools to enforce proper security headers in web server configurations to mitigate HTTP Security Headers Misconfiguration vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	XML-Related Vulnerabilities				
37	XML External Entity (XXE) Injection	XXE is a type of attack against applications that parse XML input. It occurs when an attacker can influence the processing of XML data by including external entities, which can lead to various security issues such as reading sensitive files, executing arbitrary code, or launching denial of service attacks.	Insecure XML parsers, lack of input validation, or enabling external entity processing in XML documents.	Unauthorized access to sensitive information, leakage of confidential data, server-side request forgery (SSRF), or denial of service (DoS).	Disable external entity processing, use secure XML parsers that do not resolve external entities, perform input validation and sanitization, restrict XML parser permissions, use XML firewall or web application firewall (WAF), and employ security testing techniques (e.g., fuzzing) to identify and remediate XXE vulnerabilities.
38	XML Entity Expansion (XEE)	XEE is a vulnerability similar to XXE but involves expanding XML entities excessively, leading to resource exhaustion, denial of service attacks, or crashing the XML parser.	Insecure XML parsers, enabling entity expansion, or lack of input validation in XML documents.	Resource exhaustion, denial of service (DoS), or crashing the XML parser, impacting application availability.	Limit entity expansion, configure XML parsers to restrict entity expansion, implement input validation and filtering, use XML processing libraries with built-in protections against XEE, and perform security testing to identify and mitigate XEE vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
39	XML Bomb	XML Bomb is a type of attack where an XML document with nested entities is crafted to consume excessive system resources when parsed, causing denial of service (DoS) or crashing the XML parser.	Crafted XML documents containing nested entities with recursive expansion, lack of input validation or filtering in XML parsers.	Denial of service (DoS), resource exhaustion, or crashing the XML parser, impacting system availability.	Implement XML parsing limits, restrict entity expansion depth, use efficient XML processing libraries with safeguards against XML bombs, perform input validation and sanitization, and conduct security testing to detect and mitigate XML Bomb vulnerabilities.
	Broken Access Control				
40	Inadequate Authorization	Inadequate authorization refers to the failure to properly enforce access controls, allowing unauthorized users to access sensitive resources or perform actions.	Lack of proper access control mechanisms, weak or misconfigured authorization policies, or improper implementation.	Unauthorized access to sensitive data, manipulation of critical functionalities, or exposure of confidential information.	Implement robust access control mechanisms, employ principle of least privilege, conduct regular access reviews and audits, use strong authentication methods, enforce role-based access controls (RBAC), employ centralized identity and access management (IAM) solutions, and perform security testing to identify and remediate authorization vulnerabilities.
41	Privilege Escalation	Privilege escalation occurs when an attacker gains unauthorized access to elevated privileges, enabling them to perform actions beyond their intended level of access.	Vulnerabilities in the system or application, misconfigured permissions, or exploitation of weaknesses in access controls.	Unauthorized access to sensitive data, manipulation of system configurations, or execution of malicious code with elevated privileges.	Implement least privilege principles, restrict access to sensitive resources, regularly update and patch systems and applications, use strong authentication and encryption, monitor privilege usage, conduct security awareness training, and employ security testing to identify and remediate privilege escalation vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
42	Insecure Direct Object References	Insecure Direct Object References (IDOR) occur when an application exposes internal implementation details, allowing attackers to manipulate object references and access unauthorized resources.	Lack of proper access controls, exposing internal object references, or insufficient validation of user input.	Unauthorized access to sensitive data or resources, leakage of confidential information, or manipulation of critical functionalities.	Implement proper access controls, validate and sanitize user input, enforce strict authorization checks, use indirect object references, avoid exposing sensitive information in URLs or responses, conduct security testing to identify and remediate IDOR vulnerabilities.
43	Forceful Browsing	Forceful browsing, also known as directory traversal, occurs when an attacker attempts to access restricted files or directories by manipulating URLs or other parameters.	Weak access controls, lack of input validation, or insufficient authorization checks.	Unauthorized access to sensitive files or directories, exposure of confidential data, or compromise of system integrity.	Implement secure access controls, validate and sanitize user input, enforce proper authorization checks, use secure file and directory permissions, employ web application firewalls (WAFs) or intrusion detection systems (IDS), restrict directory listing, conduct security testing to identify and remediate forceful browsing vulnerabilities.
44	Missing Function-Level Access Control	Missing Function-Level Access Control (MFAC) occurs when an application fails to properly enforce access controls on specific functions or operations.	Inadequate access control mechanisms, insufficient validation of user permissions, or improper configuration of authorization policies.	Unauthorized access to sensitive functionalities or operations, manipulation of critical system functionalities, or exposure of confidential data.	Implement granular access controls, enforce proper authorization checks for each function or operation, use role-based access controls (RBAC), conduct thorough security assessments, perform code reviews to identify and remediate MFAC vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Insecure Deserialization				
45	Remote Code Execution via Deserialization	Remote Code Execution (RCE) via Deserialization occurs when an attacker can manipulate the deserialization process of objects in an application to execute arbitrary code remotely.	Insecure deserialization implementations, lack of input validation, or allowing untrusted data to be deserialized.	Execution of arbitrary code on the server, leading to complete compromise of the system, unauthorized data access, or system takeover.	Employ secure deserialization practices, validate and sanitize input data, use libraries/frameworks with built-in protections against RCE via deserialization, employ type checking, integrity checks, and digital signatures, and restrict object instantiation and execution privileges.
46	Data Tampering	Data Tampering refers to the unauthorized modification of data, altering its integrity or structure, typically to gain an advantage or cause harm.	Weak data validation and integrity checks, lack of encryption or hashing, inadequate access controls or authentication mechanisms.	Data corruption, integrity breaches, unauthorized changes to sensitive information, financial loss, reputational damage, or system compromise.	Implement robust data validation and integrity mechanisms, encrypt sensitive data, use secure hashing algorithms, enforce access controls and authentication, implement secure communication protocols, and monitor for suspicious activities or modifications.
47	Object Injection	Object Injection occurs when untrusted data is deserialized into objects and manipulated in a way that leads to arbitrary code execution or unauthorized access to sensitive data.	Insecure deserialization practices, allowing untrusted data to instantiate objects, lack of input validation or type checking.	Arbitrary code execution, unauthorized access to sensitive information, elevation of privileges, or complete compromise of the system.	Implement secure deserialization practices, validate and sanitize input data, use libraries/frameworks with built-in protections against object injection, employ type checking and integrity checks, restrict object instantiation privileges, and limit the scope of deserialization operations.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	API Security Issues				
48	Insecure API Endpoints	Insecure API Endpoints refer to API endpoints that lack proper authentication, authorization, or encryption mechanisms.	Poorly designed or implemented API endpoints, lack of authentication or authorization checks, insufficient encryption.	Unauthorized access to sensitive data, data breaches, manipulation of resources, or exposure of sensitive information.	Implement secure authentication mechanisms (e.g., OAuth, JWT), enforce proper authorization checks, use HTTPS/TLS encryption, apply input validation and data sanitization, employ API security best practices (e.g., rate limiting, access controls), conduct regular security audits and testing.
49	API Key Exposure	API Key Exposure occurs when API keys or tokens are leaked, stolen, or exposed due to poor security practices or configuration.	Improper handling or storage of API keys, unintentional exposure in code repositories or network traffic, lack of encryption.	Unauthorized access to API resources, misuse of API services, data breaches, or compromise of user accounts.	Store API keys securely, avoid hardcoding keys in code, use secure key management practices, encrypt keys at rest and in transit, limit key access privileges, rotate keys regularly, monitor for key usage and suspicious activities, implement API key revocation mechanisms.
50	Lack of Rate Limiting	Lack of Rate Limiting refers to the absence of controls to limit the number of requests an API consumer can make within a timeframe.	Failure to implement rate limiting mechanisms, overlooking the importance of controlling API usage, insufficient security measures.	Denial of service (DoS) attacks, API abuse, resource exhaustion, or degraded system performance.	Implement rate limiting controls based on user, IP address, or API key, set appropriate rate limits for different API endpoints, monitor API usage and traffic patterns, use caching mechanisms, employ distributed denial of service (DDoS) protection, analyze and respond to abnormal usage patterns.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
51	Inadequate Input Validation	Inadequate Input Validation occurs when input data received by an API is not properly validated for correctness or security.	Lack of input validation checks, accepting and processing user-supplied data without validation, trusting input from untrusted sources.	Injection attacks (e.g., SQL injection, XSS), data corruption, security vulnerabilities, or unexpected behavior.	Implement rigorous input validation and sanitization routines, validate input data against predefined criteria and whitelists, use parameterized queries for database interactions, escape or encode output data, employ web application firewalls (WAFs) and intrusion detection/prevention systems (IDS/IPS), educate developers on secure coding practices.
	Insecure Communication				
52	Man-in-the-Middle (MITM) Attack	A Man-in-the-Middle (MITM) Attack occurs when an attacker intercepts and alters communication between two parties without their knowledge or consent.	Weaknesses in network infrastructure, lack of encryption, insecure communication channels, compromised or spoofed network devices.	Eavesdropping on sensitive data, unauthorized access or modification of data, session hijacking, impersonation, or injection of malicious content.	Implement end-to-end encryption (e.g., TLS/SSL), use secure communication protocols (e.g., HTTPS, SSH), employ digital certificates and mutual authentication, implement network segmentation and monitoring, educate users on secure browsing and communication practices.
53	Insufficient Transport Layer Security	Insufficient Transport Layer Security refers to inadequate protection of data during transmission over a network, leading to exposure to interception or tampering.	Weak encryption algorithms, outdated protocols (e.g., SSLv2, SSLv3), misconfigured encryption settings, lack of certificate validation.	Unauthorized access to sensitive data, data leakage, interception of credentials or personal information, or compromise of communication integrity.	Use strong encryption algorithms (e.g., AES), enforce the use of TLS 1.2/1.3, configure cipher suites securely, ensure proper certificate validation and revocation checks, disable insecure protocols and features, regularly update and patch systems, implement network monitoring and detection mechanisms.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
54	Insecure SSL/TLS Configuration	Insecure SSL/TLS Configuration refers to improperly configured SSL/TLS settings, leaving the communication channel vulnerable to attacks or exploitation.	Weak cipher suites, improper SSL/TLS protocol versions, lack of forward secrecy, incomplete certificate chain, weak or self-signed certificates.	Exposure of sensitive data to interception, man-in-the-middle attacks, decryption of encrypted traffic, or compromise of server integrity.	Configure SSL/TLS settings securely, use strong cipher suites with perfect forward secrecy (PFS), enable HSTS and secure renegotiation, ensure proper certificate management and validation, enable OCSP stapling, implement TLS termination at load balancers or reverse proxies, conduct regular security assessments and audits.
55	Insecure Communication Protocols	Insecure Communication Protocols involve the use of protocols that lack proper security mechanisms, making communication susceptible to interception or manipulation.	Use of outdated or insecure protocols (e.g., HTTP, Telnet), lack of encryption or authentication mechanisms, inadequate session management.	Exposure of sensitive data to interception, sniffing attacks, eavesdropping, session hijacking, unauthorized access, or manipulation of data.	Use secure communication protocols (e.g., HTTPS, SSH, SFTP), enforce encryption and authentication, implement strong session management controls, disable insecure protocols, employ VPNs or encrypted tunnels for remote access, conduct regular security assessments and vulnerability scans.
	Client-Side Vulnerabilities				
56	DOM-based XSS	DOM-based Cross-Site Scripting (XSS) occurs when an attacker injects malicious scripts into a web application's Document Object Model (DOM), leading to the execution of unauthorized code in the victim's browser.	Improper handling of user input, lack of output encoding, client-side processing of untrusted data.	Theft of session cookies, user credentials, sensitive data exposure, unauthorized actions on behalf of the user.	Implement proper input validation and output encoding, utilize Content Security Policy (CSP), sanitize user input, avoid client-side rendering of untrusted data, use security libraries and frameworks for input validation and encoding.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
57	Insecure Cross-Origin Communication	Insecure Cross-Origin Communication occurs when web applications allow communication between different origins (domains) without proper security controls, leading to data leakage or unauthorized access.	Lack of proper Cross-Origin Resource Sharing (CORS) configuration, relaxed Same-Origin Policy (SOP), insufficient authentication and authorization mechanisms.	Unauthorized access to sensitive data, leakage of confidential information, cross-site request forgery (CSRF), or manipulation of user sessions.	Implement strict CORS policies, use CSRF tokens and anti-CSRF mechanisms, enforce proper authentication and authorization controls, validate and sanitize data from cross-origin sources, minimize the exposure of sensitive APIs.
58	Browser Cache Poisoning	Browser Cache Poisoning occurs when attackers manipulate or poison the cache of a web browser, leading to the serving of malicious or unauthorized content to users.	Lack of cache validation mechanisms, improper cache-control headers, reliance on outdated or vulnerable caching mechanisms.	Delivery of malicious content, injection of unauthorized scripts or code, exploitation of user trust in cached resources.	Implement proper cache validation and cache-control headers, use secure and updated caching mechanisms, enforce HTTPS, implement strict Content Security Policy (CSP), regularly monitor and audit cache behavior.
59	Clickjacking	Clickjacking, also known as UI redress attack or User Interface (UI) manipulation, occurs when attackers trick users into clicking on hidden or disguised elements, often leading to unintended actions or malicious outcomes.	Overlaying transparent or invisible elements over legitimate content, manipulating the z-index property, exploiting vulnerabilities in browser rendering engines.	Unauthorized actions performed by users (e.g., clicking on hidden buttons), theft of sensitive information, spreading of malware or phishing attacks.	Implement frame-busting techniques, utilize X-Frame-Options header or Content Security Policy (CSP) frame-ancestors directive, employ JavaScript-based defenses (e.g., frame-killing scripts), educate users on recognizing and avoiding clickjacking attempts.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
60	HTML5 Security Issues	HTML5 Security Issues encompass vulnerabilities and risks associated with the use of HTML5 technologies, such as WebSockets, Web Storage, Geolocation API, and Web Workers, which may introduce new attack surfaces or vectors.	Lack of input validation and sanitization, improper use of browser features, insufficient security controls in HTML5 APIs.	Cross-site scripting (XSS), data leakage, privacy violations, geolocation tracking, denial-of-service (DoS) attacks.	Implement secure coding practices, validate and sanitize user input, enforce strict content security policies, utilize browser security features (e.g., sandboxing), stay informed about HTML5 security advisories and updates, conduct security assessments and penetration testing.
	Denial of Service (DoS)				
61	Distributed Denial of Service (DDoS)	Distributed Denial of Service (DDoS) attacks involve multiple compromised systems, often controlled by attackers, targeting a single system or network with a flood of traffic, overwhelming its capacity to respond to legitimate requests.	Exploitation of vulnerabilities in network protocols, misuse of legitimate network resources, exploitation of botnets or malware-infected devices.	Disruption of services, downtime, loss of revenue, damage to reputation, potential data breaches or theft.	Implement network-level protections (e.g., firewalls, intrusion detection/prevention systems), utilize rate limiting and traffic filtering mechanisms, employ DDoS mitigation services, maintain robust network infrastructure, deploy scalable and resilient architecture.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
62	Application Layer DoS	Application Layer Denial of Service (DoS) attacks target the application layer of a system or network, exploiting vulnerabilities in software or protocols to exhaust resources, degrade performance, or render services unavailable.	Insecure application design or coding, lack of input validation or sanitization, susceptibility to recursive or inefficient operations.	Degradation of application performance, unavailability of services, disruption of user experience, potential data loss or corruption.	Implement proper input validation and output encoding, use rate limiting and throttling mechanisms, employ caching and load balancing, deploy web application firewalls (WAFs), conduct security testing and code reviews, monitor and analyze application traffic.
63	Resource Exhaustion	Resource Exhaustion attacks aim to deplete system resources, such as CPU, memory, disk space, or network bandwidth, by overwhelming or consuming them with excessive requests or malicious activity.	Inefficient resource management, lack of proper resource monitoring and allocation, susceptibility to denial-of-service (DoS) attacks.	Slow performance, system crashes, unresponsiveness, service degradation, potential data loss or corruption.	Optimize resource utilization and allocation, employ rate limiting and traffic shaping, implement caching and load balancing, monitor system health and performance metrics, use resource quotas and limits, deploy robust intrusion detection and prevention systems (IDPS).
64	Slowloris Attack	Slowloris is a type of DDoS attack that aims to exhaust server resources by keeping many connections open and sending partial HTTP requests, preventing the server from serving legitimate requests and ultimately causing denial of service.	Exploitation of limitations in web server software, manipulation of HTTP connection handling, evasion of traditional DDoS mitigation techniques.	Server overload, depletion of resources, denial of service, service unavailability, disruption of website or application functionality.	Utilize web server configuration hardening, enable connection timeouts and request limits, deploy web application firewalls (WAFs), implement rate limiting and traffic shaping, use DDoS mitigation services, monitor server logs for suspicious activity.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
65	XML Denial of Service	XML Denial of Service (XDoS) attacks exploit vulnerabilities in XML parsers or processing libraries to craft specially crafted XML documents, causing resource exhaustion or excessive CPU/memory consumption during parsing or validation.	Improper handling of XML input, lack of input validation or sanitization, excessive resource usage by complex XML document structures.	CPU/memory exhaustion, application unresponsiveness, denial of service, disruption of XML-based services or applications.	Implement strict XML parsing and validation, use XML schema validation, limit XML document size and complexity, enforce timeouts and limits on XML processing, utilize XML security features and libraries, monitor XML processing performance and resource usage.
	Other Web Vulnerabilities				
66	Server-Side Request Forgery (SSRF)	Server-Side Request Forgery (SSRF) is a vulnerability that allows an attacker to manipulate server-side requests made by the web application, potentially accessing internal systems or services that should not be exposed.	Lack of input validation or sanitization, insecure handling of user-supplied input, misuse of network protocols.	Unauthorized access to sensitive internal resources, data leakage, potential data breaches, compromise of backend systems.	Implement strict input validation and sanitization, validate and restrict user-controlled input, utilize whitelisting and blacklisting, firewall and network segmentation, restrict outbound connections, employ secure coding practices and frameworks.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
67	HTTP Parameter Pollution (HPP)	HTTP Parameter Pollution (HPP) occurs when multiple values are submitted for the same parameter, leading to ambiguity or misinterpretation by the server, potentially resulting in unintended behavior or security vulnerabilities.	Lack of input validation or sanitization, improper handling of query parameters or form data, insecure coding practices.	Data corruption or manipulation, unexpected application behavior, security vulnerabilities such as injection attacks or privilege escalation.	Implement proper input validation and encoding, enforce strict parameter parsing rules, use POST requests for sensitive data, avoid using user-controlled data in dynamic queries or commands, utilize parameter separation techniques.
68	Insecure Redirects and Forwards	Insecure Redirects and Forwards occur when a web application redirects or forwards users to other URLs based on user input, without properly validating or sanitizing the redirect target, allowing attackers to conduct phishing attacks or direct users to malicious websites.	Lack of input validation or sanitization, insecure handling of redirect parameters, improper use of redirection functions.	Phishing attacks, unauthorized access to sensitive information, compromise of user credentials, installation of malware or ransomware.	Validate and sanitize redirect URLs, use whitelist-based validation, avoid dynamic redirects based on user-controlled input, employ secure coding practices, utilize secure redirection methods, implement contextual checks and access controls.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
69	File Inclusion Vulnerabilities	File Inclusion Vulnerabilities arise when a web application allows users to include files or resources dynamically, without proper validation or access controls, leading to unauthorized access, remote code execution, or data leakage.	Insecure file path handling, lack of input validation or sanitization, improper file permissions or access controls.	Unauthorized access to sensitive files or resources, remote code execution, data disclosure or leakage, compromise of server integrity.	Implement strict file path validation, restrict file inclusion to trusted directories, enforce proper file permissions and access controls, utilize whitelist-based validation, avoid user-controlled input in file inclusion functions, implement secure coding practices.
70	Security Header Bypass	Security Header Bypass occurs when security headers, such as Content Security Policy (CSP), X-Frame-Options, or X-XSS-Protection, are improperly configured or enforced, allowing attackers to bypass security controls and exploit vulnerabilities in web applications.	Misconfiguration of security headers, insecure handling of HTTP responses, lack of server-side enforcement of security policies.	Cross-site scripting (XSS) attacks, clickjacking, data injection or manipulation, bypass of security controls, compromise of user privacy or data.	Configure security headers correctly and comprehensively, enforce security policies at both client and server sides, use modern security standards and practices, perform regular security audits and testing, follow industry best practices for secure web application development.
71	Clickjacking	Clickjacking, also known as UI redressing, is a technique used by attackers to trick users into clicking on elements of a web page that are invisible or disguised, potentially leading to unintended actions or disclosures.	Insecure framing of web content, lack of frame-busting or frame-sandboxing techniques, improper handling of user interactions.	Unauthorized actions performed by the user, unintended disclosure of sensitive information, execution of malicious commands or scripts.	Implement frame-busting techniques, utilize frame-sandboxing attributes, employ clickjacking protection headers, avoid overlaying content from untrusted sources, educate users about potential risks and precautions.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
72	Inadequate Session Timeout	Inadequate Session Timeout refers to the failure to terminate user sessions after a period of inactivity, leaving sensitive session data exposed to unauthorized access or hijacking attacks, compromising user privacy and security.	Lack of session management controls, insufficient session timeout settings, insecure session token handling.	Unauthorized access to user accounts or data, session hijacking, privilege escalation, exposure of sensitive information, compromised user privacy.	Configure appropriate session timeout values based on application requirements, enforce session expiration mechanisms, implement session invalidation on user logout or browser close, use secure session tokens and cookies, conduct regular security audits and monitoring.
73	Insufficient Logging and Monitoring	Insufficient Logging and Monitoring occurs when a web application fails to log or monitor security-related events and activities adequately, making it difficult to detect and respond to security incidents or breaches in a timely manner.	Lack of comprehensive logging mechanisms, inadequate monitoring of security events, failure to implement incident response procedures.	Delayed detection of security incidents, difficulty in identifying and investigating security breaches, increased risk of data breaches or theft.	Implement robust logging mechanisms, log security-relevant events and activities, utilize intrusion detection and prevention systems (IDPS), establish centralized logging and monitoring infrastructure, conduct regular log analysis and security incident response drills.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
74	Business Logic Vulnerabilities	Business Logic Vulnerabilities stem from flaws or weaknesses in the design, implementation, or configuration of business logic processes within web applications, enabling attackers to manipulate business workflows or transactions for malicious purposes.	Insecure business logic design, lack of input validation or enforcement, improper authorization or access controls.	Financial losses, fraudulent transactions, unauthorized data access or manipulation, disruption of business operations, reputational damage.	Perform comprehensive threat modeling and risk assessment, validate and enforce business logic rules, implement proper input validation and authorization checks, conduct security testing and code reviews, monitor and audit business processes and transactions.
75	API Abuse	API Abuse occurs when attackers exploit vulnerabilities or misconfigurations in web APIs (Application Programming Interfaces) to gain unauthorized access, extract sensitive data, or perform malicious actions against web applications or backend systems.	Insecure API design or implementation, lack of authentication or authorization controls, excessive trust in client-supplied data.	Unauthorized data access or manipulation, compromised user privacy, denial of service, account takeover, exposure of sensitive information.	Implement strong authentication

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Mobile Web Vulnerabilities				
76	Insecure Data Storage on Mobile Devices	Insecure data storage on mobile devices refers to storing sensitive information (such as passwords, tokens, or personal data) in an unprotected or easily accessible manner on mobile devices.	Inadequate encryption, lack of secure storage mechanisms, storing sensitive data in plain text or weakly encrypted formats.	Unauthorized access to sensitive data, data breaches, identity theft, financial loss, regulatory non-compliance.	Implement secure data storage practices (e.g., encryption, secure key management), store sensitive data in secure storage areas (e.g., Keychain on iOS, Keystore on Android), avoid storing sensitive data locally if possible, use secure APIs for accessing data, perform security assessments and audits.
77	Insecure Data Transmission on Mobile Devices	Insecure data transmission on mobile devices refers to transmitting sensitive information over unencrypted or insecure communication channels, such as HTTP instead of HTTPS.	Lack of transport layer security (TLS), failure to enforce encryption for data in transit, reliance on insecure communication protocols.	Data interception, eavesdropping, unauthorized access to sensitive information, man-in-the-middle attacks.	Implement TLS encryption for data transmission, use HTTPS for web communication, validate server certificates, avoid transmitting sensitive data over unencrypted channels, employ secure communication libraries and APIs, enforce encryption in mobile apps, educate users about secure communication practices.
78	Insecure Mobile API Endpoints	Insecure mobile API endpoints refer to vulnerabilities in the APIs used by mobile applications, allowing attackers to exploit weaknesses and gain unauthorized access to sensitive data or resources.	Lack of input validation, insufficient authentication and authorization mechanisms, insecure data transmission, API misconfigurations.	Data exposure, unauthorized access to sensitive information, account takeover, privilege escalation.	Implement secure authentication mechanisms (e.g., OAuth, JWT), enforce authorization checks for API requests, use HTTPS for API communication, apply input validation and sanitization, employ API security best practices (e.g., rate limiting, API tokens), conduct regular security assessments of APIs, monitor and log API activity.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
79	Mobile App Reverse Engineering	Mobile app reverse engineering involves the process of decompiling, analyzing, and understanding the inner workings of a mobile application to uncover vulnerabilities, extract sensitive information, or modify its behavior.	Lack of obfuscation, weak code protection, insecure storage of sensitive data or credentials, reliance on client-side security controls.	Exposure of proprietary algorithms or intellectual property, theft of sensitive data or credentials, unauthorized app modifications.	Apply code obfuscation techniques to make reverse engineering difficult, use binary protection tools, implement runtime application self-protection (RASP), avoid hardcoding sensitive information, encrypt sensitive data, use secure authentication and authorization, conduct security testing and code reviews, monitor app behavior for suspicious activities.
	IoT Web Vulnerabilities				
80	Insecure IoT Device Management	Insecure IoT device management refers to vulnerabilities in the management interfaces or protocols of IoT devices, allowing unauthorized access, manipulation, or control of the devices and associated data.	Poorly implemented management interfaces, lack of secure authentication mechanisms, inadequate access controls, insecure communication protocols.	Unauthorized access to IoT devices, compromise of device functionality, data breaches, unauthorized data collection or manipulation.	Implement secure management interfaces with strong authentication (e.g., multi-factor authentication), enforce access controls, use secure communication protocols (e.g., TLS), regularly update device firmware, conduct security assessments and penetration testing.
81	Weak Authentication on IoT Devices	Weak authentication on IoT devices refers to vulnerabilities in the authentication mechanisms used by IoT devices, making it easier for attackers to bypass or brute-force authentication and gain unauthorized access.	Default or hardcoded credentials, lack of password policies, absence of multi-factor authentication, inadequate session management.	Unauthorized access to IoT devices, unauthorized control or manipulation of device functionality, data breaches, privacy violations.	Use strong, unique passwords for device authentication, enforce password complexity and expiration policies, implement multi-factor authentication, disable default accounts and passwords, encrypt authentication credentials, periodically review and update authentication mechanisms.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
82	IoT Device Vulnerabilities	IoT device vulnerabilities encompass various security weaknesses and flaws present in IoT devices, making them susceptible to exploitation by attackers.	Design flaws, implementation errors, insecure configurations, lack of security updates and patches, inadequate security testing.	Compromise of device integrity, unauthorized access or control, data breaches, privacy violations, disruption of services.	Regularly update device firmware and software, implement secure coding practices, conduct thorough security assessments and testing, adhere to IoT security standards and guidelines, monitor and analyze device behavior for signs of compromise, establish incident response procedures.
	Web of Things (WoT) Vulnerabilities				
83	Unauthorized Access to Smart Homes	Unauthorized access to smart homes refers to security weaknesses or vulnerabilities in smart home devices or systems that allow unauthorized individuals to gain access to the home.	Poorly configured or default settings, lack of strong authentication mechanisms, insecure communication protocols.	Unauthorized control over smart home devices, compromise of home security and privacy, potential theft or vandalism, surveillance or monitoring of occupants.	Implement strong authentication mechanisms such as multi-factor authentication, regularly update device firmware and software, use secure communication protocols (e.g., TLS), configure devices with unique and strong passwords, enable network segmentation, monitor device activity for signs of unauthorized access.
84	IoT Data Privacy Issues	IoT data privacy issues pertain to concerns regarding the protection and privacy of personal data collected and transmitted by IoT devices.	Inadequate data encryption, data leakage, insufficient user consent or control over data collection and sharing, lack of transparency in data handling practices.	Unauthorized access or disclosure of sensitive personal information, loss of privacy, potential identity theft, surveillance or profiling of individuals.	Implement data encryption for sensitive information, provide clear and transparent privacy policies, obtain explicit consent for data collection and sharing, minimize data collection and retention, anonymize or pseudonymize collected data, regularly audit data handling practices for compliance with privacy regulations.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Authentication Bypass				
85	Insecure "Remember Me" Functionality	Insecure "Remember Me" functionality refers to vulnerabilities in authentication mechanisms that allow attackers to bypass authentication requirements by exploiting flaws in the "Remember Me" feature.	Weak implementation of session management, lack of proper expiration controls for session tokens, inadequate protection of stored credentials.	Unauthorized access to user accounts, compromise of sensitive information, privilege escalation, impersonation of legitimate users.	Implement strong session management controls, enforce proper expiration policies for session tokens, encrypt and securely store user credentials, use multi-factor authentication, regularly review and audit authentication mechanisms.
86	CAPTCHA Bypass	CAPTCHA bypass involves techniques used by attackers to circumvent CAPTCHA challenges, typically used to distinguish between human users and automated bots, thereby enabling automated attacks on web applications.	Weak or outdated CAPTCHA algorithms, inadequate implementation or validation of CAPTCHA responses, reliance solely on CAPTCHA for bot detection.	Increased risk of automated attacks such as credential stuffing, account takeover, spamming, data scraping, and distributed denial-of-service (DDoS) attacks.	Implement stronger CAPTCHA mechanisms (e.g., reCAPTCHA v3), integrate CAPTCHA with other bot detection methods, implement rate limiting and IP blocking for suspicious activity, continuously monitor and update CAPTCHA solutions to mitigate emerging bypass techniques.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Server-Side Request Forgery (SSRF)				
87	Blind SSRF	Blind Server-Side Request Forgery (SSRF) is a vulnerability that allows an attacker to send crafted requests from the server to other internal or external systems.	Inadequate input validation and sanitization, lack of proper access controls, and trust in user-supplied input without proper verification.	Data leakage, unauthorized access to internal systems, server exploitation, and potential compromise of sensitive information.	Implement strict input validation and sanitization of user-controlled input, utilize allowlists for URL whitelisting, enforce proper access controls, use network-level restrictions to prevent outgoing connections to sensitive resources, and employ security tools like Web Application Firewalls (WAFs) to detect and block SSRF attacks.
88	Time-Based Blind SSRF	Time-Based Blind SSRF is a variation of SSRF where the attacker exploits delays in the server's response to infer the success or failure of SSRF attacks.	Lack of input validation and sanitization, server-side processing delays that indicate successful or unsuccessful SSRF requests, and reliance on timing-based detection mechanisms.	Similar to Blind SSRF, with the added risk of increased detection difficulty and slower exploitation, which may prolong the attacker's presence and increase the impact of the attack.	Implement server-side request timeout mechanisms to limit the time spent on processing SSRF requests, monitor server response times for anomalies, utilize anomaly detection tools to identify suspicious behavior, and conduct thorough security testing, including penetration testing and vulnerability scanning, to identify and mitigate SSRF vulnerabilities.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Content Spoofing				
89	MIME Sniffing	MIME sniffing, also known as content-type sniffing, refers to the behavior of some web browsers to override the specified MIME type of a resource and instead infer it from the content of the resource.	Lack of proper MIME type validation, browsers attempting to guess the MIME type based on the content of the resource.	Potential security risks such as content spoofing, XSS attacks, and data leakage.	Implement strict MIME type declarations in server responses, use the "X-Content-Type-Options" header with the "nosniff" directive to prevent MIME sniffing, validate user-uploaded files to ensure they match the expected MIME type.
90	X-Content-Type-Options Bypass	X-Content-Type-Options bypass refers to vulnerabilities that allow attackers to bypass the X-Content-Type-Options header, which is intended to prevent MIME type sniffing and force the browser to respect the declared content type.	Improper configuration of the X-Content-Type-Options header, allowing attackers to manipulate or circumvent the protection provided by the header.	Potential security risks such as MIME sniffing, content spoofing, XSS attacks, and data leakage.	Ensure the X-Content-Type-Options header is properly configured with the "nosniff" directive, use strong content-type validation on the server side, and employ additional security mechanisms such as Content Security Policy (CSP).
91	Content Security Policy (CSP) Bypass	Content Security Policy (CSP) bypass refers to vulnerabilities that allow attackers to circumvent or evade the protections provided by a CSP, which is designed to mitigate various types of attacks such as XSS and data injection.	Improper implementation of CSP directives, allowing attackers to inject malicious scripts or content that bypasses the CSP restrictions.	Potential security risks such as XSS attacks, data injection, unauthorized access to sensitive resources, and data exfiltration.	Implement a comprehensive CSP policy with strict directives, including proper whitelisting of trusted sources, avoiding the use of unsafe inline scripts, and regular auditing and testing of the CSP configuration for potential bypasses.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Business Logic Flaws				
92	Inconsistent Validation	Inconsistent validation vulnerabilities occur when input data is not consistently validated across different parts of an application, leading to varying levels of security checks.	Inadequate implementation of validation logic, failure to enforce consistent validation standards across all input points.	Potential security risks such as injection attacks, data tampering, privilege escalation, and unauthorized access to sensitive data.	Implement consistent validation mechanisms across all input points, use centralized validation libraries or functions, enforce strict input validation rules, and conduct thorough security testing to identify and address inconsistencies.
93	Race Conditions	Race conditions arise in multi-threaded or concurrent software systems when the sequence or timing of operations can lead to unexpected or unintended behavior.	Lack of proper synchronization mechanisms, improper handling of shared resources or critical sections, inadequate concurrency controls.	Potential security risks such as data corruption, resource contention, privilege escalation, and unauthorized access to sensitive data.	Implement proper synchronization mechanisms, use thread-safe programming practices, employ transactional controls for critical operations, and conduct thorough testing under various concurrency scenarios to detect and prevent race conditions.
94	Order Processing Vulnerabilities	Order processing vulnerabilities occur when an application's order management system is susceptible to exploitation, manipulation, or unauthorized access.	Weak or insecure authentication and authorization mechanisms, inadequate validation of order-related data, lack of proper access controls.	Potential security risks such as unauthorized order modification, data tampering, financial loss, and reputational damage.	Implement strong authentication and authorization controls, enforce strict validation of order-related data, apply role-based access controls, monitor and log order processing activities, and conduct regular security assessments and audits.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
95	Price Manipulation	Price manipulation vulnerabilities involve unauthorized modifications or alterations to product prices, often leading to financial losses or fraudulent activities.	Weak or inadequate access controls, lack of proper validation and verification of price-related data, insufficient monitoring and logging.	Potential security risks such as financial loss, revenue leakage, customer dissatisfaction, and reputational damage.	Implement strong access controls to restrict access to price-related functionalities, validate and verify price-related data inputs, monitor and log price modification activities, and implement anomaly detection mechanisms to identify suspicious price changes.
96	Account Enumeration	Account enumeration vulnerabilities occur when an attacker can determine the validity of user accounts or credentials through various techniques or methods.	Differential error messages, inconsistent response times, predictable user identifiers or account enumeration endpoints.	Potential security risks such as unauthorized access, credential stuffing attacks, brute force attacks, and account takeover.	Implement uniform error handling to avoid differential responses, ensure consistent response times for valid and invalid requests, use random or non-sequential user identifiers, and implement account lockout mechanisms to prevent brute force attacks.
97	User-Based Flaws	User-based flaws encompass a wide range of vulnerabilities that result from weaknesses or inadequacies in user-related functionalities or features within an application.	Insecure authentication and authorization mechanisms, inadequate session management, lack of user input validation, and insufficient access controls.	Potential security risks such as unauthorized access, privilege escalation, data leakage, and account compromise.	Implement strong authentication and authorization mechanisms, enforce proper session management controls, validate and sanitize user input, apply least privilege principles, and conduct thorough security testing to identify and remediate user-based flaws.

Top 100 Web Vulnerabilities					
S.No	**Vulnerability**	**Definition**	**Root Cause**	**Impact**	**Mitigation**
	Zero-Day Vulnerabilities				
98	Unknown Vulnerabilities	Unknown vulnerabilities refer to security weaknesses or flaws in software or systems that have not yet been identified or publicly disclosed.	Lack of visibility or awareness about potential security risks, limited understanding of the underlying software or system components.	Potential security risks such as unauthorized access, data breaches, system compromise, and exploitation by malicious actors.	Implement comprehensive security testing methodologies such as penetration testing, vulnerability scanning, and code reviews to identify and remediate unknown vulnerabilities. Stay updated with security advisories, patches, and updates released by software vendors and security communities.
99	Unpatched Vulnerabilities	Unpatched vulnerabilities are security flaws or weaknesses in software or systems for which no patches or updates have been applied to address the issue.	Failure to apply security patches or updates released by software vendors or maintainers, lack of patch management processes or procedures.	Potential security risks such as exploitation by attackers, data breaches, system compromise, and disruption of services.	Establish effective patch management practices to regularly identify, prioritize, and apply security patches or updates to mitigate known vulnerabilities. Implement automated patch management tools and processes to streamline the patching process and ensure timely deployment of updates.
100	Day-Zero Exploits	Day-zero exploits, also known as zero-day exploits, are attacks that target vulnerabilities in software or systems that have not yet been discovered or patched.	Lack of awareness or protection against newly identified vulnerabilities, delayed response to security advisories or vulnerability disclosures.	Potential security risks such as unauthorized access, data breaches, system compromise, and disruption of services.	Implement proactive security measures such as intrusion detection systems, behavior-based anomaly detection, and network segmentation to detect and mitigate day-zero exploits. Stay informed about emerging threats and vulnerabilities through threat intelligence feeds, security advisories, and industry forums.