Bug bounty glossary: common web application vulnerabilities

BY ELEANOR BARLOW · APRIL 23, 2025 · LAST UPDATED ON JANUARY 2, 2026

What you will learn

- **How to recognise and understand the most common web application vulnerabilities,** including broken access control, injection flaws (like SQLi and XSS), SSRF, insecure design, and outdated components, so you can prioritize what to test and fix.
- **Why each vulnerability matters in real world security testing**, including their typical impacts and examples of high profile breaches that demonstrate how they can be exploited.
- **How to mitigate these common vulnerabilities effectively**, with practical guidance on controls, secure design practices, and defensive coding to reduce risk in your web applications.

What's the difference between a risk, threat, and a vulnerability?

A **risk**, according to NIST, is defined as 'An effect of uncertainty on or within information and technology. Cybersecurity risks relate to the loss of confidentiality, integrity, or availability of information, data, or information (or control) systems and reflect the potential adverse impacts to organizational operations (i.e., mission, functions, image, or reputation) and assets, individuals, other organizations, and the Nation.'

A **threat** is a potential or actual event that is malicious or incidental. A threat can refer to any malicious or potentially malicious activity that can impact data, systems, or networks. A threat actor, also known as a threat actor or malicious actor, is an individual or group that carries out malicious activity with the intent of causing harm.

A **vulnerability** is 'a hole or a weakness in an application, source code, software, hardware, mobile, web, or infrastructure, which can be a design flaw or an implementation bug, that allows an attacker to cause harm to the stakeholders of an application. Stakeholders include the application owner, application users, and other entities that rely on the application.' - OWASP.

> **"As businesses strive to protect customer and internal data, safeguard systems, and comply with laws and regulations, understanding the impact of various vulnerabilities is essential. Each vulnerability can differ significantly in both severity and likelihood of exploitation, influenced by the specific context of the environment and any mitigation controls applied. "**
>
> **Colin Glenn, Head of Sales, Intigriti**

What this glossary covers

While there are many types of vulnerabilities, this list has been generated using the Intigriti platform to cover 20 common web application vulnerabilities observed by our Intigriti researchers. These have then been mapped to the Open World Application Security Project (OWASP), formally known as the Open Worldwide Application Security Project. This is a standard awareness document for developers and web application security, that represents a broad consensus about the most critical security risks to web applications.

Learn how these vulnerabilities work, explore real-life examples, and know what steps to put in place to safeguard against them.

Not seeing an example of a web application vulnerability that you would like more information on? Tell us!

1. Broken access control

Broken access control occurs when an application's authentication mechanism, such as login and session management, is improperly implemented, allowing users to bypass authentication controls or impersonate legitimate operators. Some common issues that contribute to broken authentication include weak passwords, insecure credential storage, session management issues, brute force vulnerability, poor implementations of Multi-Factor Authentication (MFA), account lockout issues, and exposure of authentication data.

'94% of applications were tested for some form of broken access control, with the average incidence rate of 3.81%, and has the most occurrences in the contributed dataset with over 318k.'- OWASP Top 10

Impact

The risks of broken access control or authentication include unauthorized access to user accounts, the ability to perform actions as other users, and potential compromise of entire systems, especially when administrative privileges are involved.

'Organizations faced an average cost of $2.95 million due to authentication-related breaches over a12-month period' - ***State of Passwordless Security Report, 2023***

Concept

Think about all your online accounts. Take your most visited clothing brand, for instance. To access this account, you will input a password and email address. These clothing sites might not stop users from entering unlimited wrong password attempts. This means that the hacker can make multiple attempts to gain access to the account. Since the site does not lock users out after failed attempts, eventually, the correct details are provided to gain access to the account. This is broken authentication because the website doesn't have protections like locking accounts after too many wrong attempts, in place.

Real-life example: Facebook

In 2020, the personal data of over 530 million users was exposed in an attack against Facebook. A broken authentication mechanism within Facebook's contact importer feature was exploited and used to gain

access to user accounts. Following the breach, an internal email was discovered, revealing that Facebook's strategy was to suggest that more incidents like this were common in the industry and to expect more issues like it. This shows just how prevalent these forms of attack are, and how unprepared most companies are when it comes to securing their networks.

Read more on this from the BBC, here, and view the breach timeline, here.

Mitigation in three steps

1. Implement strong password policies and ensure proper encryption.
2. Put in place multi-factor authentication (MFA) for added protection, on all devices.
3. Regularly audit and test the application for vulnerabilities in authentication.

2. Broken access control: Cross-Site Request Forgery (CSRF)

An example of broken access control is CSRF. CSRF allows threat actors to perform actions on behalf of a user, taking advantage of the user's authenticated session. The attacker exploits the fact that the user is already authenticated and logged in on the target site and then sends unauthorized requests, exploiting the user's credentials.

'Cross-site Request Forgery (CSRF) is another vulnerability in the Broken Access Control category. These vulnerabilities are widespread in modern web applications due to the use of predictable parameters for actions. However, detecting CSRF vulnerabilities is relatively straightforward through code analysis and application security testing. The identification technique involves multi-stage payload delivery and is well-documented. As a result, the severity of a CSRF vulnerability is generally considered to have an average level of exploitability.' – VeraCode.

Impact

In CSRF vulnerabilities, adversaries can perform actions on behalf of the victim without their knowledge, such as changing account settings, freezing accounts, transferring money, or posting content, leading to potential financial loss, data compromise, or reputational damage. In cases like online banking or shopping, attackers could steal sensitive personal information or trigger changes to important records, leading to data breaches.

'36% of web applications are vulnerable to CSRF attacks, underscoring the ongoing relevance of this security issue.' - Vectra.

Concept

Imagine you are at the bank. Now imagine being able to access someone else's account information if you logged in. This gives permissions to access the user profile, which in turn, gives access to open applications.

Real-life example: Netflix

Netflix fixed several flaws in its website that could allow hackers to change user addresses and hijack accounts through cross-site request forgery (CSRF). Steve Swasey, Netflix spokesperson, reported that "Netflix has 5.2 million members, and its 5.2 million members should always feel secure that we protect their private data and we take whatever measures are necessary to do that," he said. "We took whatever measures were necessary here. We always use encryption to protect our customers' credit card numbers. We take that very seriously." – SCMedia.

Mitigation in three steps

1. Implement unique tokens in forms or URLs that are tied to the user's session. This ensures that every request is legitimate and originates from the correct user.
2. Sensitive actions, such as changing passwords or making financial transactions, require users to re-enter their password or use multi-factor authentication (MFA) to confirm their identity.
3. Never access apps or accounts from links. Always go to the app/site directly.

3. Broken access control: Path traversal

'A path traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder. By manipulating variables that reference files with "dot-dot-slash (../)" sequences and their variations or by using absolute file paths, it may be possible to access arbitrary files and directories stored on the file system including application source code or configuration and critical system files.' – OWASP.

Impact

Path traversal vulnerabilities can allow attackers to access restricted files, such as system files, user data, and credentials, potentially leading to data breaches or unauthorized access to critical information. If adversaries can modify or delete files, they may cause data corruption, system failures, or operational sabotage. Additionally, accessing sensitive configuration files could enable privilege escalation, granting higher-level system access in the process. This could lead to full system compromise and exploitation of other vulnerabilities.

'94% of applications were tested for some form of broken access control with the average incidence rate of 3.81%, and has the most occurrences in the contributed dataset with over 318k.' - OWASP

Concept

Imagine you're shopping online, and you want to download your own receipt. The website gives you a link, like www. shoponline.com/files/receipts/john123.pdf You are only supposed to access your own files. Not anyone else's.

But if the site isn't properly secured, you could change the link to something like:

That ../../ is like saying "Go up two levels in the folder structure, then go into the admin folder and grab a private file." If the website doesn't check that you're allowed to access that file, it might just hand it over.

That's path traversal.

Real-life example - Fortinet FortiOS

CVE-2018-13379 refers to a vulnerability in the operating system of FortiGate firewalls. The Cybersecurity and Infrastructure Security Agency (CISA) noted this vulnerability as one of the most routinely exploited vulnerabilities in 2021 and showed how a 3-year-old vulnerability was still being used for attacks.

'The rapid shift and increased use of remote work options, such as virtual private networks (VPNs) and cloud-based environments, likely placed additional burden on cyber defenders struggling to maintain and keep pace with routine software patching.'- Read more about this, here.

Mitigation in three steps

1. Ensure that any user-provided input is properly validated and sanitized.
2. Use secure, platform-provided functions to access files.
3. Enforce strong access control to restrict who can read, write, or modify sensitive files.

4. Broken access control - Insecure direct object reference (IDOR)

IDOR is a security vulnerability used by threat actors to manipulate parameters to reference objects that they don't have permission to access. This occurs when the system is unable to validate or restrict user access. 'IDOR is a vulnerability that arises when threat actors can access or modify objects by manipulating identifiers used in a web application's URLs or parameters. It occurs due to missing access control checks, which fail to verify whether a user should be allowed to access specific data.' – OWASP.

Impact

A successful IDOR attack can lead to data breaches, privacy loss, damage to reputation, and hefty compliance penalties.

"IDOR vulnerabilities have resulted in the compromise of personal, financial, and health information of millions of users and consumers [...} hackers have used the bugs to access sensitive data, modify or delete objects, or access functions."- Cybersecurity and Infrastructure Security Agency, National Security Agency and Australian Cyber Security Centre (ACSC)

Concept

Imagine you are shopping online, but you go to the URL of the online store and change the numbers in the address. By doing this, you have been given access to someone else's order without permission. It's like leaving a door unlocked, you can peer through the door and see things you should not. In IDOR this can be done via a small error in something as simple as a number in a website link.

	Path Traversal	IDOR (Insecure Direct Object Reference)
What's exploited?	File paths in the system (e.g., using `../` to go up directories)	Object references in parameters (like user IDs or message IDs)
Focus	Gaining access to files/directories **on the server's filesystem**	Gaining access to **data or functionality** you shouldn't have
Risk area	Server file access (config files, logs, OS files)	App-level data (user accounts, messages, invoices, etc.)

Path Traversal or IDOR?

[Figure 1, Source: Intigriti, 'Path Traversal or IDOR']

Real-life example: WordPress

'The WP Private Message plugin for WordPress is vulnerable to insecure direct object reference in versions up to, and including, 1.0.6. This is due to insufficient validation on a user-controlled key identifying private messages between users. This makes it possible for authenticated attackers with subscriber-level capabilities and above to read arbitrary messages.' - Word fence.

'The WP Private Message WordPress plugin (bundled with the Superio theme as a required plugin) before 1.0.6 does not ensure that private messages to be accessed belong to the user making the requests. This allows any authenticated users to access private messages belonging to other users by tampering with the ID.'- CVE

Mitigation in three steps

1. Tighten permissions at the server level.
2. Use random or encrypted identifiers, not guessable sequences.
3. Always verify permissions before allowing access.

5. Weak cryptography

Weak cryptography refers to cryptographic algorithms or methods that are considered insecure or insufficient for protecting data in the modern digital landscape. These algorithms may have been adequate in the past, but as technology has developed, these same algorithms have since become vulnerable to various attacks.

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

Impact

Using weak cryptography can lead to data breaches, unauthorized access, and other security vulnerabilities. It's important to stay updated with the latest cryptographic standards and best practices

to ensure the security of your data.

'Organizations with weak encryption or poor cryptographic practices faced an average cost of $4.62 million per data breach' – Ponemon Institute.

Concept

Imagine you have a treasure box, and you want to keep the treasures inside safe. You use a lock to protect it. There are different types of locks, just like there are different types of cryptographic methods. A strong lock represents strong cryptography. This would be a modern padlock with a complex combination. A weak lock represents weak cryptography, which would be a simple padlock with a 3-digit combination. By using a strong lock, or strong cryptography, the safer your treasure will be.

Real-life example: LastPass

In 2020, LastPass, a password manager application, experienced a data breach due to weak cryptography practices. Threat actors were able to infiltrate and exploit vulnerabilities within the encryption system, to gain access to vaults containing sensitive data. This breach highlights the risks of poor management of encryption keys, as well as the handling of encryption methods.

'The threat actor was also able to copy a backup of customer vault data from the encrypted storage container which is stored in a proprietary binary format that contains both unencrypted data, such as website URLs, as well as fully encrypted sensitive fields such as website usernames and passwords, secure notes, and form-filled data.' – LastPass.

Mitigation in three steps

1. Use strong cryptographic algorithms.
2. Implement proper key management.
3. Stay updated and follow best practices.

6. Insufficient logging and monitoring

Insufficient logging and monitoring refer to the lack of proper tracking, recording, and reviewing of system activities or security events in an IT environment. Missed information makes systems more vulnerable to attacks, and makes it hard to detect, prevent, and respond to security threats promptly.

'Without logging and monitoring, breaches cannot be detected. Insufficient logging, detection, monitoring, and active response occurs any time' – OWASP.

Impact

When logging and monitoring are insufficient, organizations may fail to detect or respond to suspicious behavior, security breaches, or system failures promptly. This can also result in difficulty investigating an incident as the scope has not been recorded. As well as compliance violations of policies such as the General Data Protection Act (GDPR), Health Insurance Portability and Accountability Act (HIPPA), and more.

‘44% of organizations reported they had insufficient monitoring to detect breaches in real time, which often delayed their ability to respond and mitigate attacks.’- Sans Survey on Security Monitoring.

Concept

Imagine you own a store, and don’t keep track of who enters, what they buy, or when they leave. One day, someone enters, takes something, and leaves without paying. But you have no records or surveillance to identify the thief or understand how the theft happened. Insufficient logging and monitoring in a computer system works the same way. It is used to monitor and identify actions and changes in a system.

Real-life example: Snowflake

Snowflake, a data warehouse company, experienced a breach in 2024 that exposed data of over 100 organizations across industries, including finance, technology, and healthcare. A key factor that contributed to the success of this attack was insufficient logging and monitoring of access events and data patterns. The company failed to monitor internal access properly and was unable to spot unauthorized activities quickly. Not only were threat actors able to remain undetected in the network for hours but exfiltrated data without flagging security systems.

‘Snowflake was a watershed moment that signaled the significant opportunity presented by identity attacks on cloud services. It demonstrated how comparatively unsophisticated methods (logging in to user accounts with stolen credentials and dumping the data) can have the same or greater impact as a traditional network or endpoint-based cyber attack involving vulnerability exploitation, malware deployment, ransomware, etc.’ – Dan Green.

Mitigation in three steps

1. Ensure all critical systems, applications, and user actions are logged so that activities can be traced back and examined.
2. Set up automated tools to continuously monitor logs for unusual or suspicious activity. This includes setting alerts for abnormal behaviors.
3. Store logs securely and ensure they are retained for an appropriate amount of time based on regulatory requirements and business needs.

7. Security misconfiguration

A security misconfiguration is a vulnerability caused by incorrect settings in a system, application, or network. Common examples include default settings, insecure HTTP headers, unnecessary services or ports, incorrect permissions, lack of encryption, verbose error messages, misconfigured policies, and outdated software.

‘Threat agents can be an attacker with physical access to the device, a malicious app on the device that exploits security misconfiguration to execute unauthorized actions on the target vulnerable application context.’ – OWASP.

Impact

Misconfigurations can expose sensitive data, leading to breaches, financial losses, and reputational damage. They may also cause compliance violations, legal consequences, operational disruptions, and an increased attack surface, making systems more vulnerable to exploitation.

'Misconfigurations were responsible for $3.5 million in average data breach' – IBM, Cost of a Data Breach Report, 2023

Concept

Imagine you have a house with a front door. The lock mechanism in the door is misconfigured, so accepts any type of key.

Real-life example: JFSC

The Jersey Financial Service Commission (JFSC) was the target of a cyber-attack that exposed 66,806 names and addresses. The company suffered the breach in 2024 when a misconfiguration in a third-party registry system was exposed. This incident indicates how important it is to check supply chain security and properly configure systems. In response to the attack, JFSC released a statement.

'We have conducted an initial forensic review with an independent cyber security partner. This review identified that the vulnerability was due to a misconfiguration in our third-party-supplied Registry system, which had been implemented in January 2021.' Read the full statement here.

Mitigation in three steps

1. Always change default usernames and passwords to something strong and unique to you.
2. Keep your website and software up to date with the latest security patches.
3. Regularly check your systems for any security issues and fix them promptly.

8. Security misconfigurations – Exposed admin interfaces

Exposed admin interfaces refer to publicly accessible control panels and management portals within systems of networks that are only meant to be accessible to a select and approved number of users. The vulnerability allows the public, and threat actors, to gain administrative access, which can lead to compromised systems and data breaches.

'Administrator interfaces may be present in the application or on the application server to allow certain users to perform privileged activities on the site. Tests should be undertaken to reveal if and how this privileged functionality can be accessed by an unauthorized or standard user. An application may require an administrator interface to enable a privileged user to access functionality that may make changes to how the site functions.' – OWASP.

Impact

A successfully exposed admin interface can cause serious issues as it allows the user to gain access and control over systems. This can cause data breaches, manipulation, and service downtime which, in turn, can cost businesses greatly in terms of financial losses, and reputational damage.

‘Analysis of over 50 U.S. federal civilian executive branch organizations uncovered more than 13,000 distinct hosts across 100 autonomous systems, with hundreds of devices having management interfaces exposed to the public internet.’ - SecurityWeek.

Concept

Imagine you book into a hotel room, but your key card accidentally gives you access to another room, or into staff areas within the hotel you should not have permission to enter.

Real-life example: Pegasus Airlines

In a breach at Pegasus Airlines, almost 23 million files were exposed due to an exposed AWS S3 bucket, showing system software, flight processes, flight charts, navigation material, crew details, and more.

‘The company’s system administrator made a mistake and didn’t manage to properly configure the cloud environment, leaving sensitive data without password protection. The sysadmin might not have had enough training in properly configuring cloud environments and managing data, putting the company in jeopardy. Pegasus Airlines should have also had the foresight to monitor user interactions with sensitive systems and data. Had they done so, they would have noticed the improper cloud storage configuration’.- Syteca

Mitigation in three steps

1. Limit admin interface access to specific IP addresses or VPNs.
2. Implement multi-factor authentication (MFA) and strong passwords.
3. Turn off or remove any unnecessary admin interfaces.

9. Server-Side Request Forgery (SSRF)

Server-Side Request Forgery (SSRF) allows a threat actor to make HTTP requests to an arbitrary domain of the attacker's choosing. Essentially, they trick the server into making requests that it shouldn't, potentially leading to data leakage, unauthorized access, or other security issues.

‘The attacker can supply or modify a URL which the code running on the server will read or submit data to, and by carefully selecting the URLs, the attacker may be able to read server configuration such as AWS metadata, connect to internal services like HTTP enabled databases or perform post requests towards internal services which are not intended to be exposed.’ – OWASP.

Impact

A Server-Side Request Forgery (SSRF) vulnerability allows the user to map and identify vulnerable systems by forcing the server to make excessive requests. In the process, data can be accessed, leaked, and manipulated, and the internal network can be explored. This not only impacts an organization's credibility but can cost companies significantly in terms of maintaining compliance regulations.

An SSRF vulnerability, exposing data of over 100 million customers, cost Capital One an estimated $150 million in expenses, including an $80 million fine. Read more from the Office of the Comptroller of the Currency.

Concept

You have a smart home assistant like Alexa. You're outside your house, but you say, "Alexa, open the garage door." Normally, someone outside wouldn't be allowed to open it. But Alexa (the server) is inside the house and doesn't realize the command is suspicious. That's SSRF: the attacker is outside but uses a trusted internal device to access sensitive internal functions.

Real-life example – Capital One

Capital One experienced an SSRF data breach in 2019 via an exploited misconfigured Web Application Firewall (WAF). Access to AWS servers exposed the personal data of over 100 million customers, including social security numbers and bank details. What ensued was an onslaught of legal battles, lawsuits, reputation damage, and a deep dive into securing cloud configurations, performing audits, and validations.

Richard D. Fairbank, Chairman and CEO stated, "While I am grateful that the perpetrator has been caught, I am deeply sorry for what has happened...I sincerely apologize for the understandable worry this incident must be causing those affected and I am committed to making it right." Read the full statement, here.

Mitigation in three steps

1. Ensure all user-provided URLs are properly checked and sanitized to prevent malicious requests.
2. Implement network segmentation to limit the server's ability to access internal resources.
3. Allow requests only to trusted, predefined domains to prevent unauthorized access.

10. Injection: command injection

A command injection vulnerability occurs when a threat actor inserts and executes malicious commands on a server or system through an application. This happens when the user input is improperly validated, allowing the user to control the system's underlying command-line interface.

'Injection attacks, including command injection, are consistently ranked in the OWASP Top 10, affecting approximately 40% of all web applications in recent years.' – OWASP.

Impact

By exploiting this vulnerability, arbitrary commands can be run, potentially gaining unauthorized access and, from there, lifting data and damaging systems. A successful attack can leave systems compromised, disrupt services, and cause reputational damage.

'Companies that suffer a major breach like command injection often face a 20% loss in revenue' – Accenture.

Concept

You submit a homework file titled 'Science_Project.docx'. But inside the filename, you hide a command like 'Science_Project.docx && Delete_All_Grades.csv'. The teacher's grading tool isn't secure. It sees your "file name" as a system instruction, so it deletes all the grades instead of just opening your project.

Real-life example: Apache Struts

According to NIST, 'The Jakarta Multipart parser in Apache Struts 2 2.3.x before 2.3.32 and 2.5.x before 2.5.10.1 has incorrect exception handling and error-message generation during file-upload attempts, which allows remote attackers to execute arbitrary commands via a crafted Content-Type, Content-Disposition, or Content-Length HTTP header, as exploited in the wild in March 2017 with a Content-Type header containing a #cmd= string.'

'Struts is vulnerable to remote command injection attacks through incorrectly parsing an attacker's invalid Content-Type HTTP header. The Struts vulnerability allows these commands to be executed under the privileges of the Web server. This is full remote command execution and has been actively exploited in the wild from the initial disclosure.' – BlackDuck.

Mitigation in three steps

1. Ensure user input is properly validated and sanitized.
2. Use secure coding practices and avoid directly executing user input in system commands.
3. Limit privileges and update systems. Run applications with the least privileges and keep

11. Injection: SQL Injection

Structured Query Language (**SQL) injection** occurs when a threat actor manipulates a website or application's Structured Query Language. This is typically done by inserting a malicious SQL code into an input field, such as a search bar, login form, or URL parameter, which is then executed by the database. When the application does not properly validate or sanitize user input, users can use this vulnerability to interact with the underlying database.

'A successful SQL injection exploit can read sensitive data from the database, modify database data (Insert/Update/Delete), execute administration operations on the database (such as shutdown the DBMS), recover the content of a given file present on the DBMS file system and in some cases issue commands to the operating system.' - OWASP

Impact

SQL injection can result in data breaches, data manipulation, authentication bypass, system compromise, and associated reputational and legal damage.

'In the case of a SQL attack, the cost per record compromised can range from $150 to $200 on average.' – IBM, Cost of a Data Breach Report

Concept

Imagine a website that asks for your username and password to log in to your account. When you type in your username and password, the website checks if the information is correct. But an attacker can try to trick the system and edit the form, by entering something special in the username or password fields. This makes the database think the threat actors' "login" is valid because the system is tricked into always saying "yes". The threat actor can now access the system without knowing the real password.

For example:

Username: admin' --

Password: *(leave blank)*

This creates: SELECT * FROM users WHERE username = 'admin' --' AND password = '';

The double dash -- starts a comment in SQL, which tells the database to ignore everything after it (including the password check).

Real-life example: Talk Talk

In 2015 TalkTalk, the telecom group, experienced an SQL attack that led to the insertion of code on the website's search functions, which led to the exposure of the financial data of 150,000 customers. The consequence of the breach cost the company a £400,000 fine for not protecting customer data.

Read more about the attack from the BBC, here.

Mitigation in three steps

1. Separate SQL logic from user input using parameterized queries.
2. Whitelist expected input types and formats.
3. Leverage secure database abstraction tools to avoid manual SQL.

12. Injection: XML External Entity (XXE)

XML, also known as XXE, is a common format used to store and transport data in many applications, but if the XML parser is not securely configured, a threat actor can inject an external entity into the XML document. These external entities are references to external files or resources, which can then be accessed or loaded by the application.

‘This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server-side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.’ – OWASP

Impact

XXE vulnerabilities can lead to several issues of high severity. They can include unauthorized data exposure, where access to sensitive files like system configurations or passwords is made. They can also trigger Denial of Service (DoS) attacks, overwhelming the server and causing disruptions. Attackers may exploit XXE to perform Server-Side Request Forgery (SSRF), tricking the server access of internal systems or databases. In some cases, XXE vulnerabilities can enable remote code execution, leading to full system compromise and unauthorized control over the application.

‘The average cost of IT downtime is $5,600 per minute. Because there are so many differences in how businesses operate, downtime, at the low end, can be as much as $140,000 per hour, $300,000 per hour on average, and as much as $540,000 per hour at the higher end.’ – Gartner

Concept

Think of the online banking app you use. You upload a document to this app as part of its verification process. The app reads the document's contents using XML to process your information. However, a threat actor creates a specially crafted XML file that tricks the app into accessing sensitive information stored on the bank's server, like account details or user passwords, and sends it back to the attacker. This is possible because the app didn't properly protect against XXE attacks, allowing the user to steal sensitive data without the bank being alerted.

Real-life example: Java Applications

‘Since most Java XML parsers have XXE enabled by default, this language is especially vulnerable to XXE attack, so you must explicitly disable XXE to use these parsers safely.’- Medium

Mitigation in three steps

1. Ensure that the XML parser used by your application is configured to disable the processing of external entities.
2. Select XML parsers that are secure by default and do not permit the processing of external entities or use libraries that automatically address such security issues.
3. Implement strict validation and sanitization of all XML inputs to ensure only expected and safe data is processed.

13. Injection: Cross-Site Scripting (XSS)

XSS allows threat actors to run harmful code on other browsers. Attackers inject malicious scripts (usually JavaScript) into web pages that are viewed by other users. These scripts can execute in the context of the victim's browser, leading to a variety of harmful outcomes.

‘Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.’ - OWASP

Impact

XSS attacks can lead to significant security risks, by capturing sensitive information like login credentials and personal details, as well as session hijacking through the theft of cookies to impersonate victims. Threat actors can also distribute malware by redirecting users to harmful websites or initiating downloads.

‘Web application attacks (which include XSS) accounted for 22% of all data breaches in 2023’ – Verizon Data Breach Report 2023

Concept

Imagine you're reading comments on a popular online forum. One comment seems normal, and you go to click on it to read more. But hidden inside is a malicious script that runs when you click on it. This script steals your username and password. The threat actor can then use this to log in and access your account.

Real-life example: eBay

‘eBay is in the headlines once again this week as the online auction site has **reportedly** been compromised by a cross-site scripting (XSS) attack, in which users were redirected to a spoof site designed to steal their credentials. This latest attack follows an **announcement** from the company back in May urging its users to change their passwords after one of its databases containing encrypted passwords and other customer data had been compromised via a small number of employee log-in credentials, allowing unauthorized access to eBay’s corporate network’. - LRQA

Mitigation in three steps

1. Ensure all user inputs are properly sanitized.
2. Implement a strong content security policy to control which sources of content.
3. When displaying user input on the website, always escape special characters to prevent them from being interpreted as code by the browser.

14. Vulnerable and Outdated Components

When software vulnerabilities remain unaddressed through updates or patches, attackers can exploit these weaknesses to gain unauthorized access, deploy malware, or disrupt services. This can result in data breaches, financial losses, and reputational damage for both individuals and organizations.

‘Vulnerable Components are a known issue that we struggle to test and assess risk and is the only category to not have any Common Vulnerability and Exposures (CVEs) mapped to the included CWEs, so a default exploits/impact weight of 5.0 is used.’ – OWASP

Impact

Old components can compromise system performance and stability, as outdated versions may conflict with newer technologies or become incompatible with essential security tools. As cyber threats evolve, failing to patch software leaves systems defenseless against emerging attack vectors, making regular updates crucial for safeguarding against potential exploitation.

By not updating components companies risk not complying with regulations. GDPR fines can go as high as €20 million or 4% of global annual turnover, whichever is higher. Read more on fines and penalties, here.

Concept

Imagine you have a house. The house has some great alarm systems, locks on the door, and fingerprint scanning as you enter. But this house also has a basement where a window is cracked. The broken window is a vulnerability because it allows intruders to easily enter your home. If you don't fix (patch) the broken window, it remains a security risk.

Real-life example: Sing Health

The data of 1.5 million patients was exposed due to vulnerabilities in outdated systems within Sing Health, a Singapore-based healthcare provider. The key issue that led to the breach was a lack of security updates in legacy components.

'Singapore's July 2018 personal data breach of 1.5 million Sing Health patients, including Prime Minister Lee Hsien Loong, was caused by threat system management, a lack of employee training, and other major flaws.' – TechTarget

Mitigation in three steps

1. Keep all software and systems up to date with the latest security patches.
2. Use tools to automatically apply updates and patches to your systems.
3. Regularly conduct security audits to identify and fix vulnerabilities.

15. Identification and Authentication Failures

Identification and authentication failures occur when a system fails to enforce proper restrictions on who can access specific resources, data, or functions. Essentially, it happens when unauthorized users or applications can access or perform actions on parts of the system, they shouldn't have permission to.

'Confirmation of the user's identity, authentication, and session management is critical to protect against authentication-related attacks.' – OWASP

Impact

Improper access control can lead to several serious issues, including unauthorized access to sensitive data, privilege escalation, fraud, identity theft, and reputation damage. It may also result in regulatory

non-compliance, risking legal consequences and fines.

'Organizations are spending an average of approximately $3 million on activities relating to authentication failures annually.' – Ponemon Institute

Concept

You walk up to an intercom on a resident block of apartments, or a shared working space building. You ring random people and say "Hey, it's Amy from XX, I left my key/pass just inside the door, can you buzz me in" and they do it without doing any ID check.

Real-life example: Equifax

While authentication failures weren't the root cause of the breach, authentication mechanisms could have played a role in mitigating the impact of the exploit that occurred in the 2017 Equifax data breach. Equifax had failed to put in place security patches to Apache Struts, a web application framework. The exploitation allowed access to sensitive data of around 147 million users.

'Less than 24 hours after Equifax revealed that all of its customers' personal info had been plundered by hackers, the company is facing a multi-billion-dollar consumer lawsuit, and an investigation by at least one state attorney general.' – Chris Mills

Mitigation in three steps

1. Grant users the minimum level of access they need to perform their job.
2. Continuously monitor and audit user access permissions to ensure they are accurate.
3. Use strong authentication methods (like multi-factor authentication) to ensure users are who they say they are and ensure that access control mechanisms (such as role-based access control) are properly configured.

16. Identification and Authentication Failures: Brute force

A brute force attack is where threat actors attempt to access a system by trying every possible combination of passwords, keys, and codes until the correct one is found. Computational power is used to test every option.

'A web application can be attacked via brute force by taking a word list of known pages, for instance from a popular content management system, and simply requesting each known page then analyzing the HTTP response code to determine if the page exists on the target server.' – OWASP

Impact

A successful brute force attack can result in compromised accounts, data, and systems. Data leaks can result in financial loss, downtime, and reputation damage.

‘Every year, cloud accounts that are compromised via brute force attacks cost companies an average of over $6 million’ – Okta

Concept

In front of you sits a safe. It is not your safe, so you don’t know the code. To unlock it, you must provide a 4-digit combination. You try every possible option until you find the right one. A brute force attack is the same, only it uses computation power to speed up the lengthy process.

Real-life example: Dunkin’ Donuts

Between 2015 and 2018, the sweet treat company Dunkin’ Donuts received a series of brute force and credential-stuffing attacks. Stolen passwords and usernames were collected from previous breaches and used to access app accounts of customers. The company failed to notify customers quickly or update security measures, which resulted in a lawsuit that led to a $650,000 settlement.

"Dunkin' failed to protect the security of its customers," Attorney General Letitia James said in a statement. "And instead of notifying the tens of thousands impacted by these cybersecurity breaches, Dunkin' sat idly by, putting customers at risk."- **wbur**

Mitigation in three steps

1. Enforce complex passwords.
2. Require a second layer of security, such as a text message code or authentication app, to verify user identity.
3. After a certain number of failed login attempts, temporarily lock the account or introduce delays to slow down the attacker's attempts.

17. Insecure design

Insecure design refers to flaws or weaknesses in the architecture or design of a system that make it vulnerable. It occurs when security considerations are not properly integrated, leaving systems exposed to exploitation.

‘Insecure design is a broad category representing different weaknesses, expressed as “missing or ineffective control design.” An insecure design cannot be fixed by a perfect implementation as by definition, needed security controls were never created to defend against specific attacks. One of the factors that contribute to insecure design is the lack of business risk profiling inherent in the software or system being developed, and thus the failure to determine what level of security design is required.’ - OWASP

Impact

Insecure design can lead to data breaches, downtime, and financial losses, but it can also cause non-compliance-related legal consequences with standards and regulations.

‘Insecure APIs and bot attacks have collectively cost global firms between $35 billion and $87 billion’ - Foresiet

Concept

You visit a hotel. To enter via the front of the building you have to use a keycard system to gain entry. But, walk a little around the building, and the back doors are wide open. Anyone can go in through those doors. The hotel should secure all entry points, but because of insecure design, they have not. The same goes for insecure design in cyber.

Real-life example: Target Data Breach

In 2013, Target was the subject of a data breach that disclosed 40 million credit and debit card numbers. Attackers entered through a third-party HVAC vendor with weak credentials and then pivoted into Target’s internal payment systems. There was no network segmentation between vendor systems and point-of-sale systems.

This is an example of Insecure Design, as the system trusted third-party access too much, and lacked proper isolation between critical infrastructure. This is a failure at the design level, not just implementation.

Mitigation in three steps

1. Implement security measures and threat assessments early in the design phase to identify and address vulnerabilities.
2. Perform continuous security testing, such as code reviews, penetration testing, and vulnerability assessments, to catch design flaws and weaknesses.
3. Ensure developers follow secure coding guidelines and best practices, including input validation, proper authentication, and access control mechanisms.

18. Insecure design – Insecure data storage

Insecure data storage refers to a lack of security measures in place to keep sensitive information such as PPI data, secure.

‘Insecure data storage in a mobile application can attract various threat agents who aim to exploit the vulnerabilities and gain unauthorized access to sensitive information. These threat agents include skilled adversaries who target mobile apps to extract valuable data, malicious insiders within the organization or app development team who misuse their privileges, state-sponsored actors conducting cyber espionage, cybercriminals seeking financial gain through data theft or ransom, script kiddies utilizing pre-built tools for simple attacks, data brokers looking to exploit insecure storage for selling personal information, competitors and industrial spies aiming to gain a competitive advantage, and activists or hacktivists with ideological motives’ - OWASP

Impact

Insecure data can result in identity theft, financial fraud, and data breaches, damaging both individuals and businesses. It also exposes organizations to legal consequences, reputational harm, and operational disruptions, which can be detrimental to finances and operations.

'A poorly secured data haven is an open invitation for cybercriminals'. According to Statista, 'the global average cost per data breach was 4.45 million U.S. dollars in 2023' – Data Storage Insights

Concept

A restaurant prints your full credit card number on your receipt and tosses the carbon copy into the rubbish without shredding it. Anyone digging through the bin can now steal your card info.

Real-life example: Facebook

Over 540 million records were exposed when Facebook data was left vulnerable through unsecured cloud storage.

'Facebook clarified that the data was not directly leaked from its own systems but was the result of improper security practices by app developers who used Facebook's APIs to collect user data. Facebook reportedly worked to notify the third-party developers and encouraged them to fix the security vulnerabilities. It also restricted access to the API that allowed apps to collect such data, making it harder for future data leaks to occur due to misconfigurations.' – CloudTech

Mitigation in three steps

1. Convert sensitive data into a secret code that can only be read with a specific key or password.
2. Restrict who can view or modify the data by setting permissions, so only authorized users or systems can access sensitive information.
3. Keep secure copies of important data.

19. Software and data integrity failures

Software and data integrity failures refer to the alteration or corruption of data that leads to errors or vulnerabilities.

They 'relate to code and infrastructure that does not protect against integrity violations. An example of this is where an application relies upon plugins, libraries, or modules from untrusted sources, repositories, and content delivery networks (CDNs). An insecure CI/CD pipeline can introduce the potential for unauthorized access, malicious code, or system compromise.'- OWASP

Impact

Integrity failures can lead to corrupted or compromised data and systems, and open the door for large-scale cyber-attacks which, in turn, can cause financial loss, legal repercussions, downtime, and loss of customer trust.

'Data breaches caused by system flaws or corruption cost companies an average of '$3.86 million per incident.' – IBM Cost of a Data Breach Report, 2020

Concept

You log into an app and track your activity on that app. Some activities are missing and some are duplicated. The app is not verifying data correctly, which leads to mistakes on your account. The data entered has been altered inappropriately, and decisions are based on incorrect information.

Real-life example: T-Mobile

In 2021 T-Mobile experienced a data breach, exposing the data of over 40 million customers. The cause came down to a weakness in the system design. Threat actors were able to access data by manipulating how it was accessed and stored, which led to data failure in data integrity and unauthorized access.

'We are currently in the process of informing impacted customers that after a thorough investigation, we have determined that a threat actor used a single Application Programming Interface (or API) to obtain limited types of information on their accounts.' – read the full statement, **here**.

Mitigation in three steps

1. Restrict access to sensitive data and systems using strict authentication and authorization measures.
2. Use automated tools to continuously monitor and validate data.
3. Ensure all software and systems are regularly updated with security patches to fix vulnerabilities.

20. Software and data integrity failures - Insecure deserialization

Insecure deserialization occurs when untrusted data is converted from a format, like JSON, XML, or binary, into an object without proper validation. Threat actors can exploit this by sending malicious data that, when deserialized by the system, can trigger harmful actions.

'Many programming languages offer a native capability for serializing objects. These native formats usually provide more features than JSON or XML, including customizability of the serialization process. Unfortunately, the features of these native deserialization mechanisms can be repurposed for malicious effects when operating on untrusted data. Attacks against deserializers have been found to allow denial-of-service, access control, and remote code execution (RCE) attacks.'- **OWASP**

Impact

Insecure deserialization can result in Remote Code Execution (RCE), Denial of Service (DoS), privilege escalation, and system compromise.

'300% increase in deserialization attacks over three months' - Imperva

Concept

Imagine you have a safe where you store all your important documents. You trust that only safe packages will be placed inside this container. However, one day, you place a package that appears safe but, unbeknown to you, contains a broken, leaking pen. Once inside, the ink spreads across your documents and damages the safe. This scenario represents insecure deserialization, where a malicious object (leaking pen), is hidden within a package that appears safe but spreads once in the targeted environment.

Real-life example: Apache Struts

In 2027 the web application framework, Apache Struts, was breached in an attack that allowed threat actors to execute code on vulnerable servers. This was due to insecure desterilization.

'A Java XStream deserialization vulnerability in the Apache Struts REST plugin, affecting Struts versions 2.1.2 to 2.3.34, and 2.5.x to 2.5.13. Java deserialization vulnerabilities occur when Java applications deserialize user-supplied data without sanitization. Successful exploitation of CVE-2017-9805 can grant the attacker remote code execution (RCE) ability on the vulnerable server.' - Gigamon

Mitigation in three steps

1. Use secure serialization formats that provide built-in mechanisms for validating and processing data safely.
2. Avoid using custom or unsafe serialization formats that may be vulnerable.
3. Use techniques like digital signatures or cryptographic hashes to verify the integrity and authenticity of serialized data before deserializing it.

Intigriti helps customers reliably and consistently identify vulnerabilities across their dynamic attack surface. A dedicated and highly skilled team triages vulnerabilities, considers their environment, validates them, and then provides them to the customer's team for prioritizing remediation or logging and accepting risk.

For more information on bug bounty terminology, contact the team here.

AUTHOR

Eleanor Barlow

Eleanor Barlow is a London-based Senior Cyber Security Technical Writer at Intigriti, with 9+ years' experience reporting on and writing for the cyber and tech sector. She specializes in data-driven content on cybersecurity and bug bounty intelligence, helping organizations benefit from the latest trends and insights.