Top 5 Web Application Security Vulnerabilities

All types of cyber attacks are rising exponentially across the globe, but there’s been a particular boost in targeted web attacks in the last few years.

Believe it or not, 17% of all cyber attacks target vulnerabilities in web applications. And if that wasn't worrying enough for organisations, 98% of typical web applications are vulnerable to these attacks. So, it's no surprise that web application security vulnerabilities are a huge talking point among cyber security enthusiasts!

If you're new to the cyber security world, you might be trying to learn more about the top web application security vulnerabilities. While SQL injections, broken access control, and authentication failures might sound unfamiliar, we’re here to make these concepts crystal clear in just a few minutes.

It doesn’t matter whether you're an aspiring red team hacker or want to learn more about preventing data breaches as part of an internal security team—you’re bound to find something useful in this detailed rundown!

Top Web Application Security Vulnerabilities

1: Broken Access Control

Access control ensures that users can *only* access files and documents directly related to their role and work. By having adequate access control in place, you risk unauthorised intrusion on sensitive documents (and increase digital security as a result).

If access control is implemented correctly, the chances of a data breach reduce significantly *without* impacting business information silos that keep an organisation ticking.

If you don’t have adequate measures in place, you risk the following things happening:

· Elevation of privilege that allows hackers to gain access to admin accounts

· Metadata manipulation

· Confidential information, system data, and user data leaking

· Unauthorised individuals making changes to system data that may disrupt critical day-to-day activities

· URL manipulation that gives hackers unauthorised access to sensitive data

Most organisations implement a Principle of Least Privilege (PoLP) policy to reduce risk. This cyber security concept involves giving employees the minimum access level possible to do their jobs. It typically prevents them from accessing confidential information, and access can constantly be updated if necessary.

Organisations can also determine what access rights should be granted or revoked over time by carrying out regular access control audits. Revoking rights (where necessary!) should reduce the organisation's overall attack surface by minimising potential access points.

2. SQL Injection

An SQL injection is when cyber attackers use malicious SQL code to manipulate a backend database and gain access to hidden information (or information that wasn't intended to be displayed).

The leaked information can include sensitive company data, intellectual property, customer details, and login credentials.

There are several common types of SQL injections, with the most common being:

1. **Error-based SQL injection**: In these scenarios, an attacker will send SQL queries to a database that causes it to display error messages. This process gives the attackers more information that they can use to extract sensitive information from the database. The best way to circumvent this type of injection is to disable error messages once a web application has officially gone live.

2. **Inferential SQL Injection**: This is often called a blind SQL injection and involves a malicious actor sending data payloads to a database through a Boolean Injection or a Time-Based Injection. Boolean Injections force a database or application to send a result, while Time-Based Injections rely on waiting a set time before a response is sent from the database.

3. **Out-of-Band Injection**: This is a relatively rare version of an SQL injection, but it's a prime method for attacking slower servers. In these scenarios, an attacker can't usually use the same channel to launch an attack and receive results. So, the extracted information is obtained through a *different* connection than what they initially attacked.

Preventing SQL Injection attacks is all about filtering database inputs, restricting code where possible, and restricting database access with least-privilege methods. It's also important to intervene at the initial stages of an application's life by checking input validation and using parameterised queries with clear parameters that can prevent breakthroughs.

3. Identification and Authentication Failure

One of the leading web application security vulnerabilities most businesses face is identification and authentication failure. This process involves identifying a user *uniquely* and is usually done with a username and password.

Whether it's while logging into internal systems or when a customer is trying to access their account to purchase from an e-commerce site, identification issues can crop up at any point.

Attackers will usually try any of the following things to access a system and take advantage of authentication failure:

- Brute forcing their way into a system by trying a collection of password combinations OR using automated tools to find valid credentials
- Take advantage of weak password rules that allow users to choose easy-to-guess passwords
- Uncover documents that store login credentials in plain text (believe it or not, these are often stored in system files!)
- Finding URLs that contain session IDs
- Social media account squatting to mimic a genuine account

To tackle identification and authentication failure, businesses need to consider setting up anti-automation controls, multi-factor authentication, and adequate security training for internal employees.

Enforcing an excellent password policy is also critical, as a staggering 73% of users have the same password for multiple sites.

4. Cryptographic Failures

Cryptography is the process of hiding or disguising coded data to ensure that *only* those who are meant to see the code can decipher it. Encrypting data scrambles any cleartext into ciphertext, which should appear unreadable to anyone without a key. So, sensitive data remains locked down.

However, many organisations can experience cryptographic failures if any of the following things occur:

- Employees send data in clear text and use HTTP to access web applications. In case you weren't aware, HTTPS is the secure version of HTTP (*anyone* can read what's sent over an insecure HTTP connection!)
- Data is protected with weak encryption or isn't masked during transit
- Businesses rely on a weak cryptographic algorithm that is easy to break in the event of a hack
- There's insecure password management

While cryptographic failures are worrying, you *can* mitigate them. It's a great idea to use excellent encryption keys and convert any plain text passwords into cipher text.

However, organisations should also ensure web application developers follow secure coding practices and conduct regular penetration testing. By understanding the weaknesses in a cryptographic approach, businesses can prioritise fixes and patch vulnerabilities in record time.

5. Cross-Site Scripting

Cross-site scripting (XSS) attacks are becoming more common. They involve injecting malicious scripts into trusted websites. By doing this, attackers attach code that loads instead of what's intended. Attackers usually craft a link encouraging a user to click, but a script can also force-post your cookies directly to the cyber criminal.

This is one of the web application security vulnerabilities that happens when HTML tags return to a client. And it's a *huge* issue, as the infected JavaScript that comes with cross-site scripting puts a user's information seriously at risk.

Not only can this technique be used to seize login credentials, but it can also uncover private information like credit card details (all without the site owner knowing a breach has occurred!). In addition, malicious JavaScript can read and modify browsers, gain access to webcams and microphones, and even impersonate users by accessing cookies. So, it can be a *huge* issue!

To prevent malicious actors from doing this to a website, you'll need to do the following things:

- Use Burp Suite to scan websites for security vulnerabilities
- Encode all data on output

·	Validate any input that comes *through*

·	Train and maintain awareness among staff to make sure no one clicks any suspicious links

·	Creating headers within HTML documents to prevent scripts from loading on alternative domains