OWASP Top 10 Cheat Sheet

1. **Injection**. Injection flaws, such as SQL, NoSQL, OS, and LDAP injection, occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data can trick the interpreter into executing unintended commands or accessing data without proper authorization.

2. **Broken Authentication**. Application functions related to authentication and session management are often implemented incorrectly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities temporarily or permanently.

3. **Sensitive Data Exposure**. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser.

4. **XML External Entities (XXE)**. Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

5. **Broken Access Control**. Restrictions on what authenticated users are allowed to do are often not properly enforced. Attackers can exploit these flaws to access unauthorized functionality and/or data, such as access other users' accounts, view sensitive files, modify other users' data, change access rights, etc.

6. **Security Misconfiguration**. Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

7. **Cross-Site Scripting XSS**. XSS flaws occur whenever an application includes untrusted data in a new web page without proper validation or escaping, or updates an

existing web page with user-supplied data using a browser API that can create HTML or JavaScript. XSS allows attackers to execute scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites.

8. **Insecure Deserialization**. Insecure deserialization often leads to remote code execution. Even if deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks.

9. **Using Components with Known Vulnerabilities**. Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.

10. **Insufficient Logging & Monitoring**. Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.

A1. Injection

- When data is sent to a website it should be sanitised.
- In vulnerable systems untested data is used by the system as part of a command or query.
- The attacker's malicious data could trick the programme interpreter, the command, or query, into running unintended commands or to allow access to data without proper authorisation.

SQL (Structured Query Language) is a language for interacting with, and managing data in a relational DB. Popular databases you have probably heard of (or used) include MySQL, Oracle, Microsoft SQL Server and PostgreSQL.

Web applications are designed for nice and friendly users. They're not always designed to protect themselves against attackers, especially on internal, corporate applications where it is assumed everyone is a nice friendly trustworthy employee etc.

SQL injection is a powerful database attack where an attacker can do pretty much anything with and to the data in a database that is exposed to a vulnerable field via a website.

Even though its existence has been known about for a long time, along with ways to mitigate the threat, there is still a large amount of vulnerable website online, which is why it is still the number one threat as report by the OWASP.

A2. Broken Authentication

- Incorrectly implemented authentication and session management can lead to situations where passwords are exposed, left as defaults, or can be re-used or bypassed.
- If a website uses Session IDs in the URL:
  - http://example.com/sale/saleitems?sessionid=2P0OC2JSNDLPSKHCJUN2JV
- An authenticated user of the site may post the URL to friends on social media without realising they are also giving away the Session ID
  - Anyone following the link will use the same Session ID, thus exposing sensitive user account data
- If a user visiting that website simply closes the browser window without logging out, and Session IDs do not time out after a period of inactivity, then any other user of the same browser will still be logged in if they also visit that website
- If the website doesn't use strong hashing processes to store user passwords, a data breach will expose those passwords to the attackers

A3. Sensitive Data Exposure

- Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII.
- Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes.
- Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

A4. XML External Entities (XXE)

- All major browsers have an inbuilt XML Parser
- Many older or poorly configured XML processors evaluate external entity references within XML documents.
- External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks.

- An attacker could send malicious code to the server parser with the intention of remotely accessing files

A5. Broken Access Control

- Restrictions on what authenticated users are allowed to do are not properly enforced.
- Attackers can exploit these flaws to access unauthorized functionality and/or data, such as
    - access other users' accounts
    - view sensitive files
    - modify other users' data
    - change access rights, etc.

A6. Security Misconfiguration

- Many websites still use default or outdated security settings, the security environment is constantly changing and security configuration needs to be monitored and updated accordingly.
- Security misconfiguration is a very commonly seen issue, mainly due to:
    - insecure default configurations
    - sensitive data in banners and headers
    - incomplete or ad hoc configurations
    - open cloud storage
    - misconfigured HTTP headers
    - verbose error messages containing sensitive information.
- Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched/upgraded in a timely fashion.

A7. Cross-Site Scripting (XSS)

- XSS attacks take advantage of your browser's trust in the data being sent by the web server.

- In a XSS attack an attacker is able to post data onto a website that will later be executed by the victim's browser.
- Cross Site Scripting XSS attacks exploit vulnerabilities in dynamically generated web pages. The common types of XSS are:
    - Server Side (aka Persistent or Stored)
    - Client Site (aka Non-persistent or reflective)
    - DOM-based
- Fields on a website can be configured to contain script, as well as images or URLs, which, when visited by a client will run the code of an attackers choice.

A8. Insecure Deserialization

- Serialisation is a way that developers turn their data structures into a stream of bytes for transport or storage
    - Deserialization is the reverse process that happens when the data is received
- The problem is that there's no way to know what you're deserializing before you've decoded it
- So, an attacker can serialise a bunch of malicious objects and send them to your application
    - Once you call readObject(), it's too late
- The attacker's malicious objects have already been instantiated, and have taken over your entire server

A9. Using Components with Known Vulnerabilities

- Components, such as libraries, frameworks, and other software modules, run with the same privileges as the application.
- If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover.
- Applications and APIs using components with known vulnerabilities may undermine application defenses and enable various attacks and impacts.
- Programmers often download pre-written code for their applications instead of writing their own
- If any code has a vulnerability in it, they automatically expose the web service to attack

A10. Insufficient Logging & Monitoring

- Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to
  - Further attack systems
  - Maintain persistence
  - Pivot to more system
  - Tamper, extract, or destroy data
- Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring.