## Web Application Security

* **Webserver Security**
* **Hash** – function that is to be/can be applied to a string.  Is any function that can be used to map data of arbitrary size to data of fixed size. The values returned by a **hash** function are called **hash** values, **hash** codes, digests, or simply **hashes**.

**MD5** - is a widely used hash function producing a 128-bit hash value. Although MD5 was initially designed to be used as a cryptographic hash function, it has been found to suffer from extensive vulnerabilities.

**SHA -** (Secure Hash Algorithm 1) is a cryptographic hash function designed by the United States National Security Agency and is a U.S. Federal Information Processing Standard published by the United States NIST.

* **Salted Hash -** Salts are used to safeguard passwords in storage. Historically a password was stored in plaintext on a system, but over time additional safeguards developed to protect a user's password against being read from the system. A salt is one of those methods.
* **Cross-site Request forgery -** also known as one-click attack or session riding and abbreviated as CSRF(sometimes pronounced sea-surf) or XSRF, is a type of malicious exploit of a website where unauthorized commands are transmitted from a user that the web application trusts.
* **Cross- site scripting -** is a type of computer security vulnerability typically found in web applications. XSS enables attackers to inject client-side scripts into web pages viewed by other users. Across-site scripting vulnerability may be used by attackers to bypass access controls such as the same-origin policy.

**OWASP**

The Open Web Application Security Project (OWASP) is open communities dedicated to enabling organizations to develop, purchase, and maintain applications and APIs that can be trusted.

Here are the lists of the OWASP Top 10 – 2017:

1. Injection – Injection flaws, such as SQL, OS, XXE, 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 and Session Management – 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. 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 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.
4. Broken Control Access – 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.
5. Security Configuration – Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, platform, etc. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.
6. 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.
7. Insufficient Attack Protection – The majority of applications and APIs lack the basic ability to detect, prevent, and respond to both manual and automated attacks. Attack protection goes far beyond basic input validation and involves automatically detecting, logging, responding, and even blocking exploit attempts. Application owners also need to be able to deploy patches quickly to protect against attacks.
8. Cross-Site Request Forgery (CSRF) – A CSRF attack forces a logged-on victim’s browser to send a forged HTTP request, including the victim’s session cookie and any other automatically included authentication information, to a vulnerable web application. Such an attack allows the attacker to force a victim’s browser to generate requests the vulnerable application thinks are legitimate requests from the victim.
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 defences and enable various attacks and impacts.
10. Unprotected APIs – Modern applications often involve rich client applications and APIs, such as JavaScript in the browser and mobile apps that connect to an API of some kind (SOAP/XML, REST/JSON, RPC, GWT, etc.). These APIs are often unprotected and contain numerous vulnerabilities

References:

MD5. (n.d.). Retrieved from <https://en.wikipedia.org/wiki/MD5>

SHA-1. (n.d.). Retrieved from <https://en.wikipedia.org/wiki/SHA-1>

The Ten Most Critical Web Application Security Risks. (2017). Retrieved from <https://www.owasp.org/images/3/3c/OWASP_Top_10_-_2017_Release_Candidate1_English.pdf>