**Injection**

Code injection consists on sending untrusted data to an interpreter (SQL, OS, LDAP) as part of a command or query. This might trick the interpreter into executing an arbitrary command in order to access or alter data without proper authorization.

This method is easy to exploit and can make use of diverse interpreter queries (SQL, LDAP, XPath, NoSQL), OS commands, XML parsers, or SMTP headers among others, which makes it one of the most prevalent web application attacks. Some of the most common types of injection are:

-SQL Injection:

This injection attack abuses the input data entry process from the client to the application in order to insert or ‘inject’ SQL commands to dynamically construct a SQL query that will be executed by the target system. By executing SQL arbitrary commands, the attacker might tamper with the target database in multiple ways, such as reading, altering or deleting sensitive data.

-Blind SQL Injection:

This attack is a specific type of SQL injection where the attacker asks true or false questions (Boolean queries) to the database, and then takes advantage of the application’s generic error messages to determine the answer. This method is used by attackers to retrieve information from web applications that don’t output data to the web page.

-XPath Injection / Blind XPath Injection:

The procedure for this attack is identical to that of an SQL injection, except that it’s used for web applications that request user-supplied information to construct an XPath query for XLM data. A blind variant that uses Boolean queries exists.

-Code Injection:

This attack injects code to be directly interpreted by the application, such as PHP code. It exploits a lack of proper input/output data validation.

-Command Injection:

This attack is used to execute arbitrary commands on the host operating system, when a vulnerable application passes user-supplied data such as forms or cookies to a system shell.

**Broken Authentication and Session Management**

Authentication and session management functions are often not implemented correctly, presenting flaws in areas such as logout, password management or timeouts. An attacker might abuse these leaks to access passwords, keys or session tokens, or impersonate users.

**Cross-Site Scripting (XSS)**

This attack takes advantage of web applications where the output makes use of the user input without validating or encoding it. The attacker might then inject malicious scripts into a trusted web site to create a malicious URL, and use social engineering to trick a victim into visiting such an URL. The user’s browser will assume the script comes from a trusted source and will execute it. This might compromise any sensitive information retained by the browser, such as cookies or session tokens.

-Stored XSS Attacks

Also known as Persistent or Type-I XSS. The script is injected directly into a database, message forum or similar, in a way that it is permanently stored on the target server. When a user requests the stored information, that user will also inadvertently retrieve the malicious script.

-Reflected XSS Attacks

Also known as Non-Persistent or Type-II XSS. The script is sent to the server as part of the input of a request. A vulnerable server might then include the script in the output, such as in an error message or search result. Social engineering is used to lure the victims into clicking a link or submitting a specific form in such a way that the injected script is reflected back to them as an output.

**Insecure Direct Object References**

A direct object reference refers to the process of exposing an internal implementation object, such as a file or database key. If incorrectly implemented, any attacker might access unauthorized data by manipulating those references. For instance, without proper access control check, an attacker might access any account by modifying the ‘account’ parameter in their browser’s address.

**Cross-Site Request Forgery (CSRF)**

This attack requires the user to be authenticated. Then, making use of social engineering such as sending a link or abusing image tags, the attacker might trick the user into executing an unintended action. Once authenticated, the web application has no way to differentiate between a genuine request and a forged one. While this might not be used to steal data, since the attacker doesn’t see the response to the forged request, it might force the victim to execute actions such as transferring funds or deleting an account.

**Clickjacking**

Multiple transparent or opaque layers are used to trick the user into unintentionally clicking on a button or link, which might redirect them to a malicious site or force them to execute an undesired action.

**Denial of Service (DoS)**

The attacker floods the target system with a large number of requests, making it unavailable for legitimate users. Alternatively, the attacker may instead exploit a programming vulnerability to achieve the same effect.

**Man-in-the-middle**

The attacker intercepts a communication between two systems. For instance, http transactions normally involve a TCP connection between client and server, but an attacker could split that connection into two (client-attacker and attacker-server), allowing the attacker to read or modify the exchanged data.

**Man-in-the-browser**

This attack follows the same approach as the man-in-the-middle attack, but it makes use of a Trojan Horse as an interceptor between the browser and its security mechanisms. It bypasses authentication factors.

**Brute Force Attack**

The attacker makes requests to a server testing every possible value of a parameter, analyzing the responses until a successful value is found. While the attacker might try every possible combination, it’s common to instead try for specific sets of values such as a dictionary attack or rainbow tables.

**Social Engineering**

Social engineering refers to the practice of taking advantage of human psychology in order to access confidential information, instead of exploiting flaws in software. It could be divided into multiple categories:

-Baiting:

The attacker leaves a malware-infected physical device such as a USB flash drive in a place where the victim is likely to find it, usually with a curiosity-inciting label. The unsuspecting victim might pick up the device and plug it into their computer, installing the malware.

-Phishing:

The attacker impersonates a reputed source and sends an email to the victim, trying to trick the recipient into sharing personal or confidential information, or clicking a malicious link. When the target is a specific individual or organization, it receives the name of ‘spear phishing’.

-Pretexting:

The attacker will make use of a pretext (an invented scenario or elaborate lie) to manipulate the victim into divulging confidential information, such as a scammer pretending to need certain information pertaining to their target in order to confirm their identity.

-Scareware:

Scareware is a type of malware that attempts to trick victims into downloading malicious software. Scareware might disguise itself as system messages coming from an antivirus or firewall application, and falsely inform about a number of inexistent problems, suggesting purchasing or downloading actual malicious software to fix the problems.