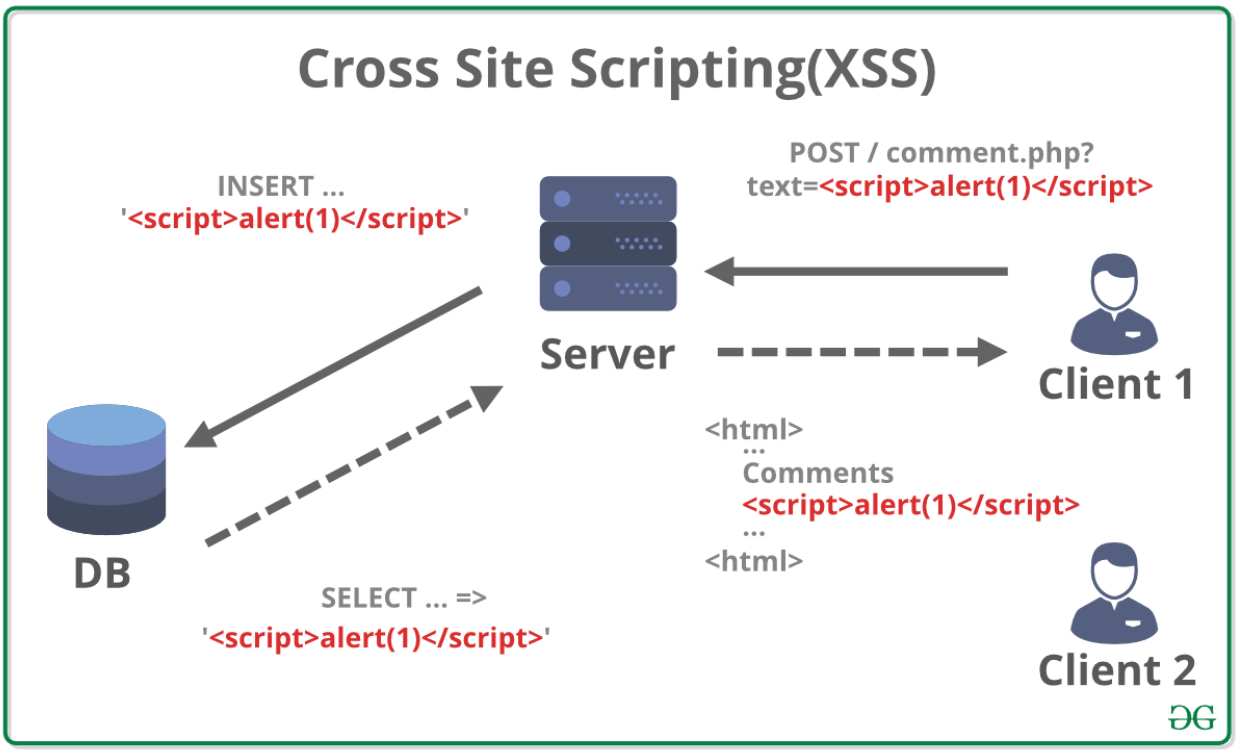
Cross Site Scripting (XSS) is a vulnerability in a web application that allows a third party to execute a script in the user's browser on behalf of the web application. Cross-site Scripting is one of the most prevalent vulnerabilities present on the web today. The exploitation of XSS against a user can lead to various consequences such as account compromise, account deletion, privilege escalation, malware infection and many more.

XSS is the most common security vulnerability in software today. This should not be the case as XSS is easy to find and easy to fix. XSS vulnerabilities can have consequences such as tampering and sensitive data theft.



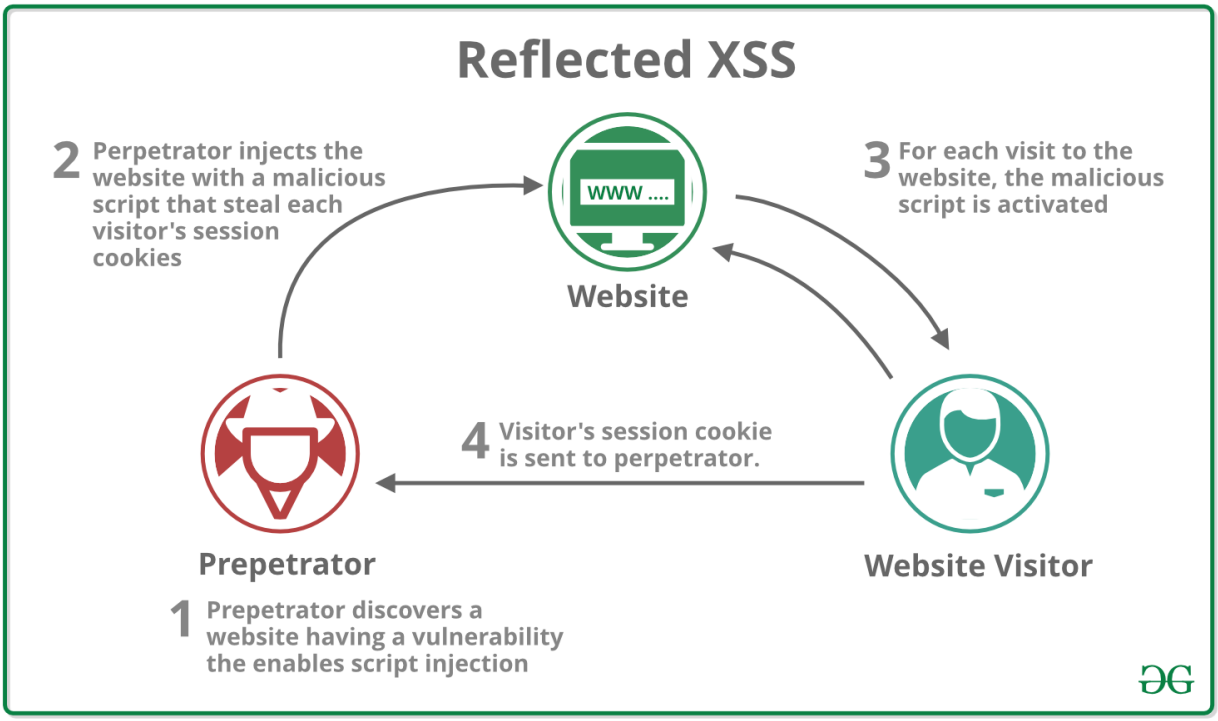
Key Concepts of XSS

- XSS is a web-based attack performed on vulnerable web applications.
- In XSS attacks, the victim is the user and not the application.
- In XSS attacks, malicious content is delivered to users using JavaScript.

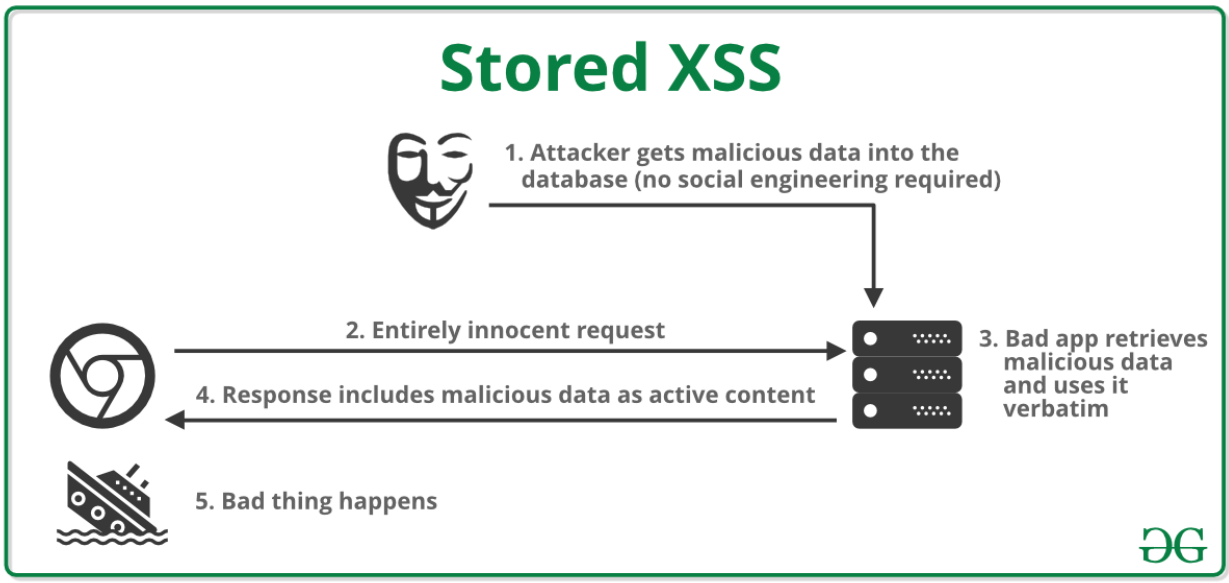
The possibility of getting XSSed arises when a website does not properly handle the input provided to it from a user before inserting it into the response. In such a case, a crafted input can be given that when embedded in the response acts as a JS code block and is executed by the browser.

Depending on the context, there are ***two types*** of XSS –

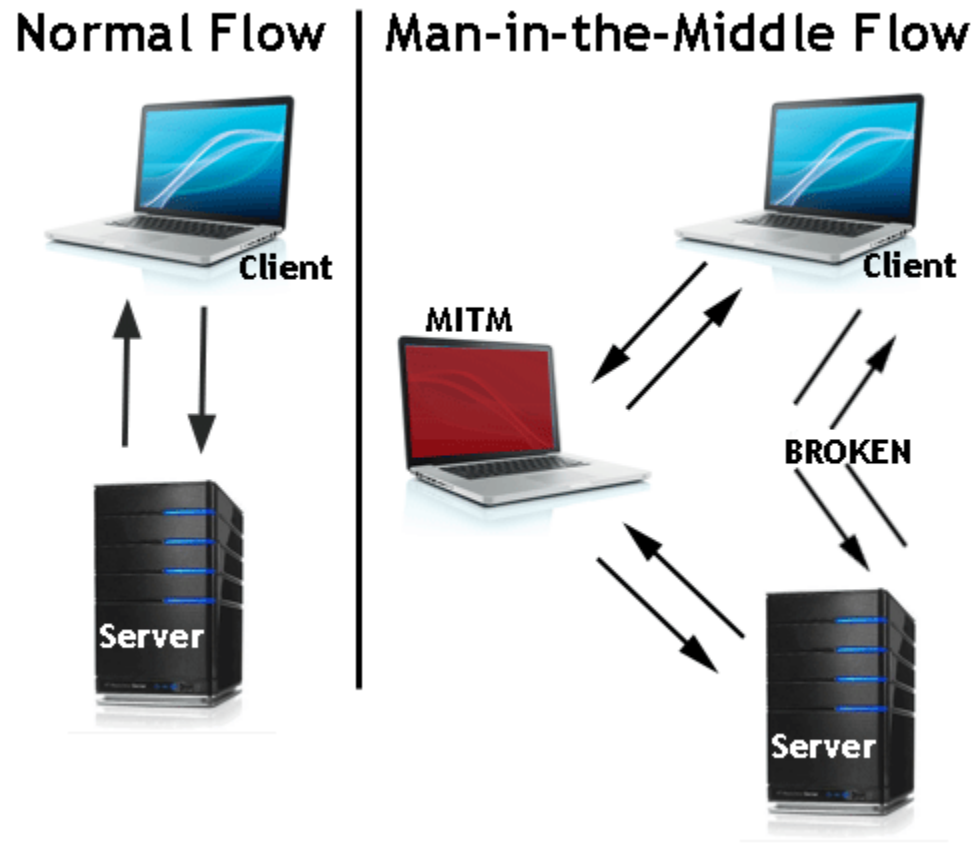
1. **Reflected XSS:**
   If the input has to be provided each time to execute, such XSS is called reflected. These attacks are mostly carried out by delivering a payload directly to the victim. Victim requests a page with a request containing the payload and the payload comes embedded in the response as a script. An example of reflected XSS is XSS in the search field.

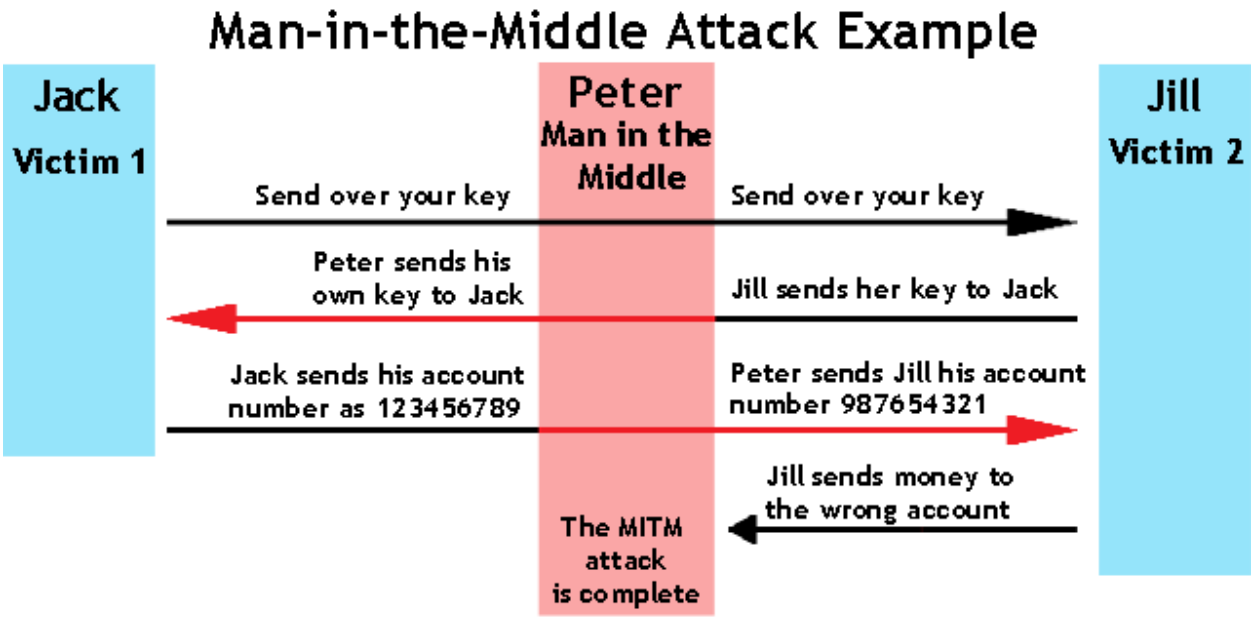


2. **Stored XSS:**
   When the response containing the payload is stored on the server in such a way that the script gets executed on every visit without submission of payload, then it is identified as stored XSS. An example of stored XSS is XSS in the comment thread.



M**AN In the Middle Attack:**





The hacker is impersonating both sides of the conversation to gain access to funds. This example holds true for a conversation with a client and server as well as person-to-person conversations. In the example above, the attacker intercepts a public key and with that can transpose his own credentials to trick the people on either end into believing they are talking to one another securely.

Interactions Susceptible to MITM Attacks

- Financial sites – between login and authentication
- Connections meant to be secured by public or private keys
- Other sites that require logins – where there is something to be gained by having access

SQL INJECTION:

SQL Injection is an attack that poisons dynamic SQL statements to comment out certain parts of the statement or appending a condition that will always be true. It takes advantage of the design flaws in poorly designed web applications to exploit SQL statements to execute malicious SQL code.