Security and Vulnerability

WEB security

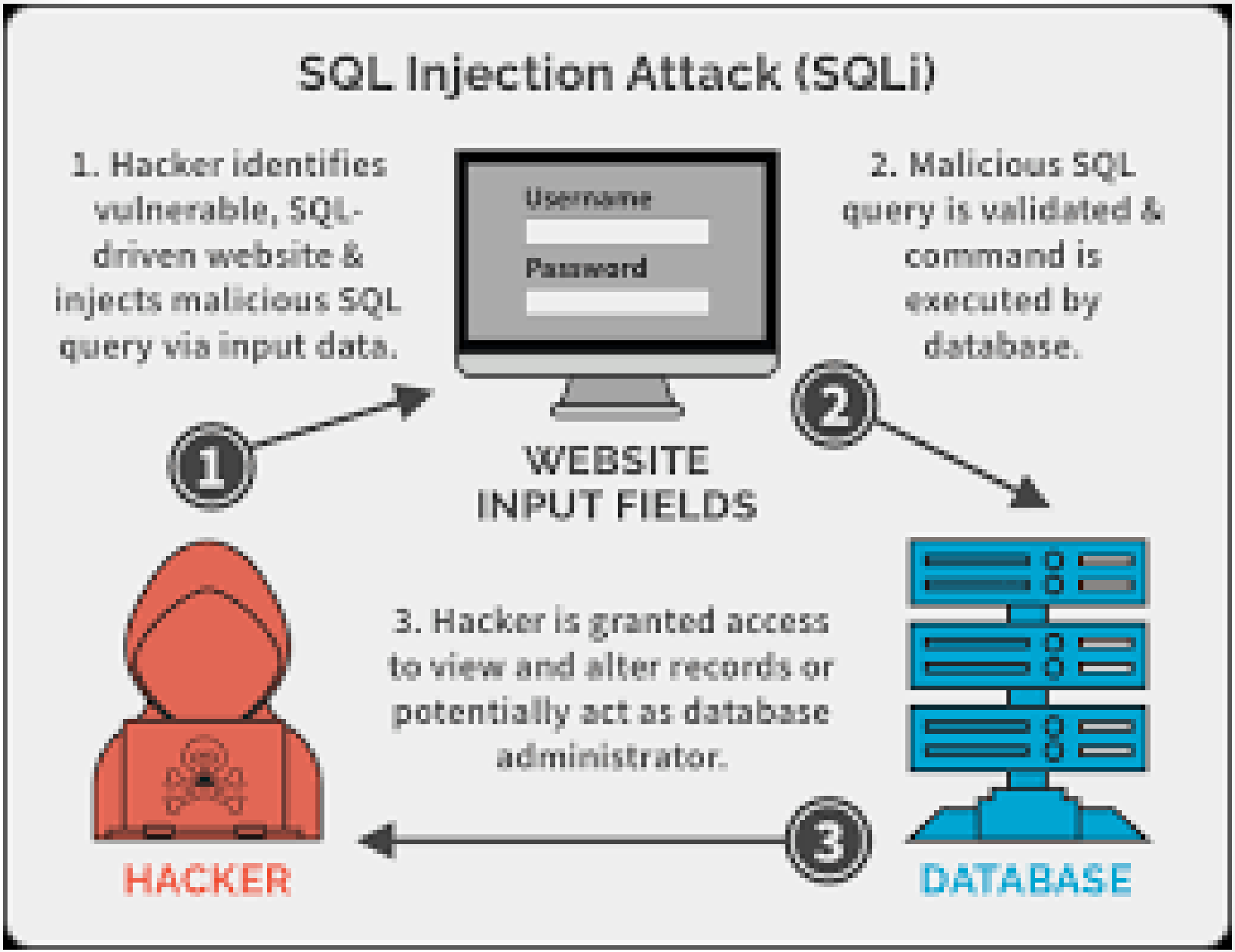
Top Web security threat

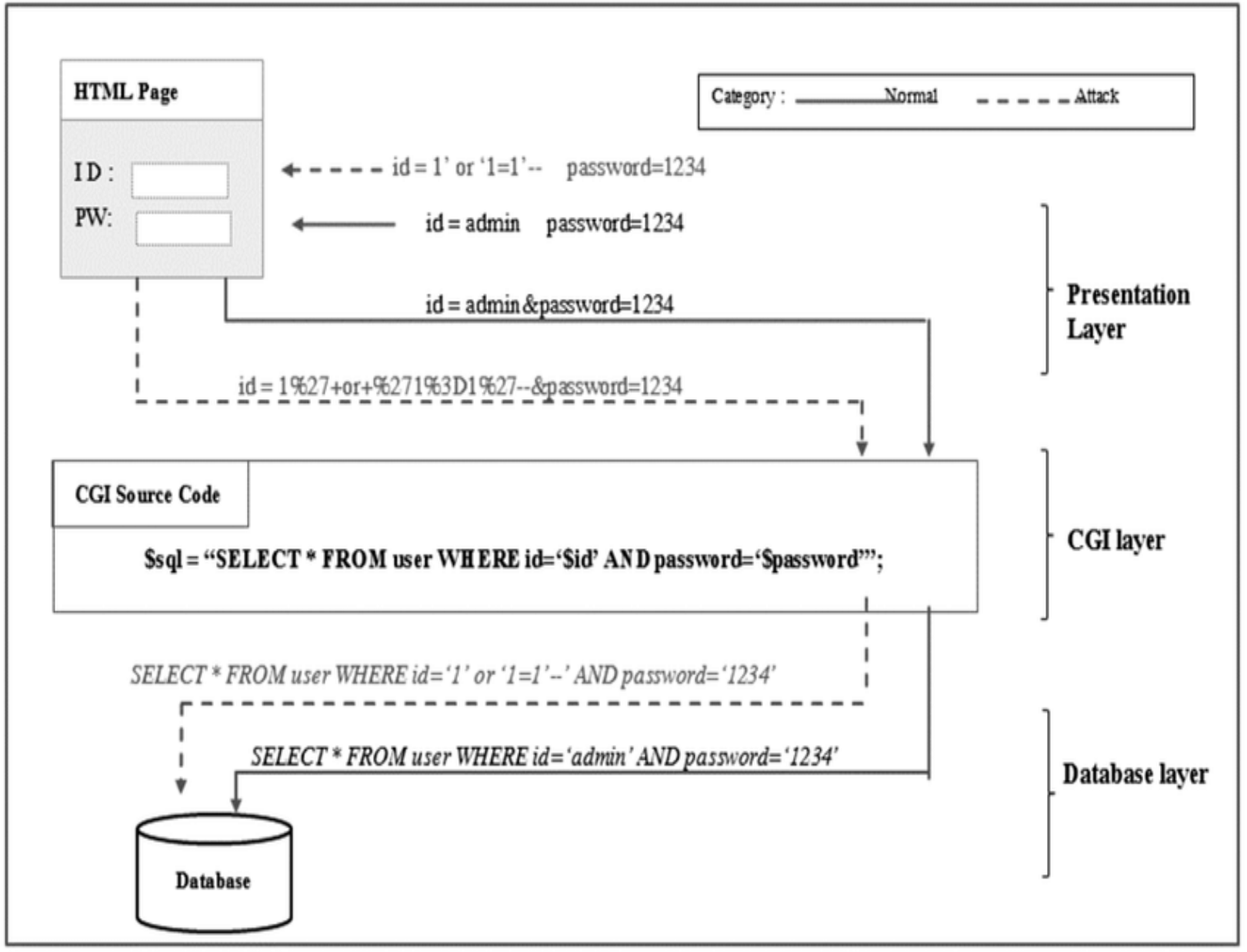
- SQL Injection
- Cross Site Scripting(XSS)
- Broken Authentication and Session Management
- Insecure Direct Object References
- Cross Site Request Forgery
- Security Misconfiguration
- Insecure Cryptographic Storage
- Failure to restrict URL Access
- Insufficient Transport Layer Protection
- Invalidated Redirects and Forwards

1. SQL injection

- Injection is a security vulnerability that allows an attacker to alter backend SQL statements by manipulating the user supplied data.
- Injection occurs when the user input is sent to an interpreter as part of command or query and trick the interpreter into executing unintended commands and gives access to unauthorized data.
- The SQL command which when executed by web application can also expose the back-end database.
- **Implication**
  - An attacker can inject malicious content into the vulnerable fields.
  - Sensitive data like User Names, Passwords, etc. can be read from the database.
  - Database data can be modified (Insert/Update/ Delete).
  - Administration Operations can be executed on the database
- **Vulnerable Objects**
  - Input Fields
  - URLs interacting with the database.

SQL Injection Attack (SQLi)



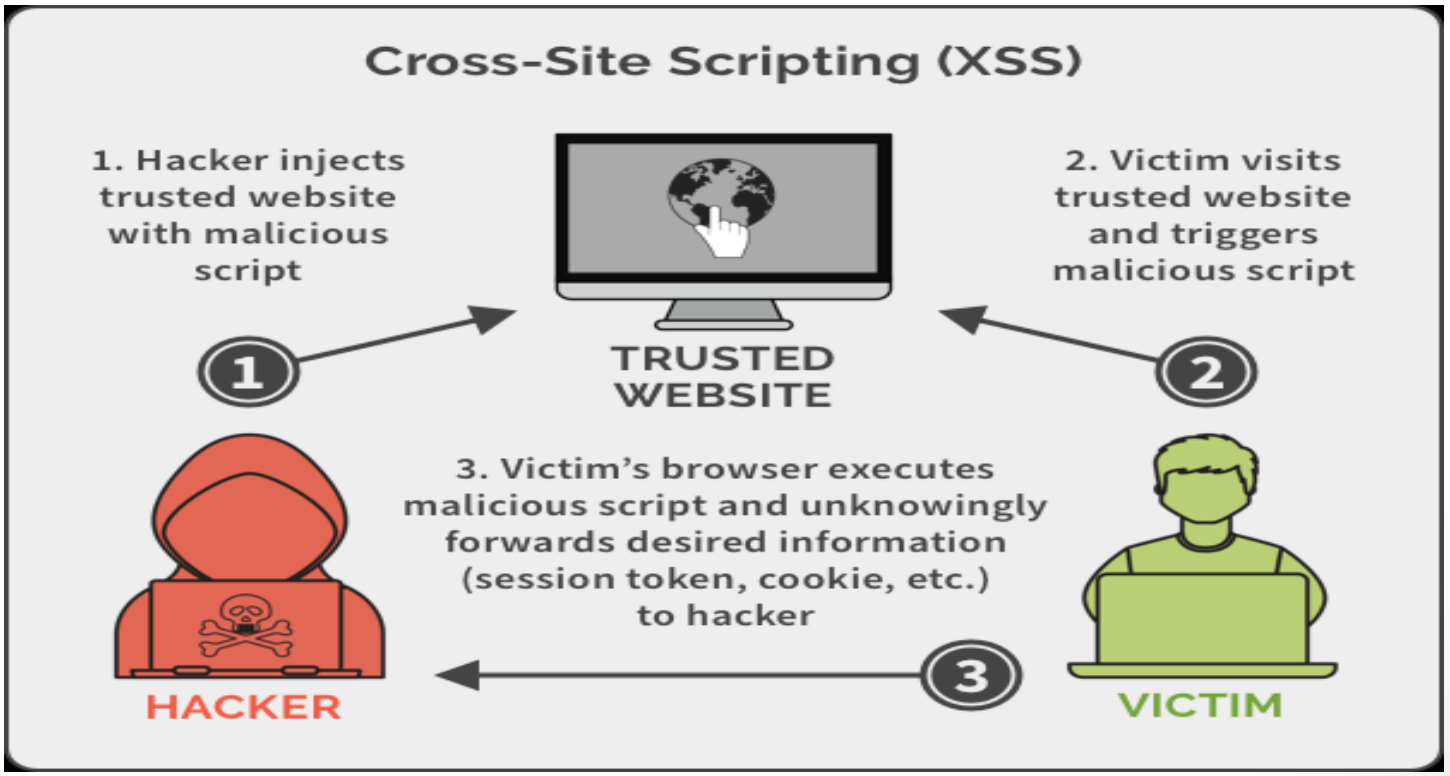

HTML Page
ID :
PW:
Category : Normal Attack
id = 1' or '1=1'-- password=1234
id = admin password=1234
id = admin&password=1234
id = 1%27+or+%271%3D1%27--&password=1234
Presentation Layer
CGI Source Code
$sql = "SELECT * FROM user WHERE id='$id' AND password='$password'";
CGI layer
SELECT * FROM user WHERE id='1' or '1=1'--' AND password='1234'
SELECT * FROM user WHERE id='admin' AND password='1234'
Database
Database layer

SQL injection on the Login Page

- Logging into an application without having valid credentials.
- Valid userName is available, and password is not available.
  - Test URL: **http://demo.testfire.net/default.aspx**
  - User Name: sjones
  - Password: 1=1' or pass123
  - SQL query created and sent to Interpreter as below
  - SELECT * FROM Users WHERE User_Name = sjones AND Password = 1=1' or pass123;
- **Recommendations**
  - White listing the input fields
  - Avoid displaying detailed error messages that are useful to an attacker.

2. Cross Site Scripting (XSS)

- XSS vulnerabilities target scripts embedded in a page that are executed on the client side i.e. user browser rather then at the server side.
- These flaws can occur when the application takes untrusted data and send it to the web browser without proper validation.



- An attacker can inject malicious content into the vulnerable fields.
- Sensitive data like User Names, Passwords, etc. can be read from the database.
- Database data can be modified (Insert/Update/ Delete).
- Administration Operations can be executed on the database

- **Vulnerable Objects**
  - Input Fields
  - URLs interacting with the database.

3. Broken Authentication and Session Management

- The websites usually create a session cookie and session ID for each valid session, and these cookies contain sensitive data like username, password, etc. When the session is ended either by logout or browser closed abruptly, these cookies should be invalidated i.e. for each session there should be a new cookie.
- If the cookies are not invalidated, the sensitive data will exist in the system. For example, a user using a public computer (Cyber Cafe), the cookies of the vulnerable site sits on the system and exposed to an attacker. An attacker uses the same public computer after sometime, the sensitive data is compromised.
- In the same manner, a user using a public computer, instead of logging off, he closes the browser abruptly. An attacker uses the same system, when browses the same vulnerable site, the previous session of the victim will be opened. The attacker can do whatever he wants to do from stealing profile information, credit card information, etc.
- A check should be done to find the strength of the authentication and session management. Keys, session tokens, cookies should be implemented properly without compromising passwords.

- **Vulnerable Objects**
  - Session IDs exposed on URL can lead to session fixation attack.
  - Session IDs same before and after logout and login.
  - Session Timeouts are not implemented correctly.
  - Application is assigning same session ID for each new session.
  - Authenticated parts of the application are protected using SSL and passwords are stored in hashed or encrypted format.
  - The session can be reused by a low privileged user.

- **Recommendations**
  - All the authentication and session management requirements should be defined as per OWASP Application Security Verification Standard.
  - Never expose any credentials in URLs or Logs.
  - Strong efforts should be also made to avoid XSS flaws which can be used to steal session IDs.

- **Implication**
  - Making use of this vulnerability, an attacker can hijack a session, gain unauthorized access to the system which allows disclosure and modification of unauthorized information.
  - The sessions can be high jacked using stolen cookies or sessions using XSS.

Example

- Airline reservation application supports URL rewriting, putting session IDs in the URL:**http://Examples.com/sale/saleitems;jsessionid=2P0OC2oJM0DPXSNQPLME34SERTBG/dest=Maldives** (Sale of tickets to Maldives)
- An authenticated user of the site wants to let his friends know about the sale and sends an email across. The friends receive the session ID and can be used to do unauthorized modifications or misuse the saved credit card details.
- An application is **vulnerable to XSS**, by which an attacker can access the session ID and can be used to hijack the session.
- Applications **timeouts are not set properly**. The user uses a public computer and closes the browser instead of logging off and walks away. The attacker uses the same browser some time later, and the session is authenticated.

4. Insecure Direct Object References

- It occurs when a developer exposes a reference to an internal implementation object, such as a file, directory, or database key as in URL or as a FORM parameter.

- The attacker can use this information to access other objects and can create a future attack to access the unauthorized data.

- **Implication**
  - Using this vulnerability, an attacker can gain access to unauthorized internal objects, can modify data or compromise the application.
- **Vulnerable Objects**
  - In the URL.
- **Examples:**
- Changing "userid" in the following URL can make an attacker to view other user's information.
  - **http://www.vulnerablesite.com/userid=123** Modified to **http://www.vulnerablesite.com/userid=124**
- An attacker can view others information by changing user id value.

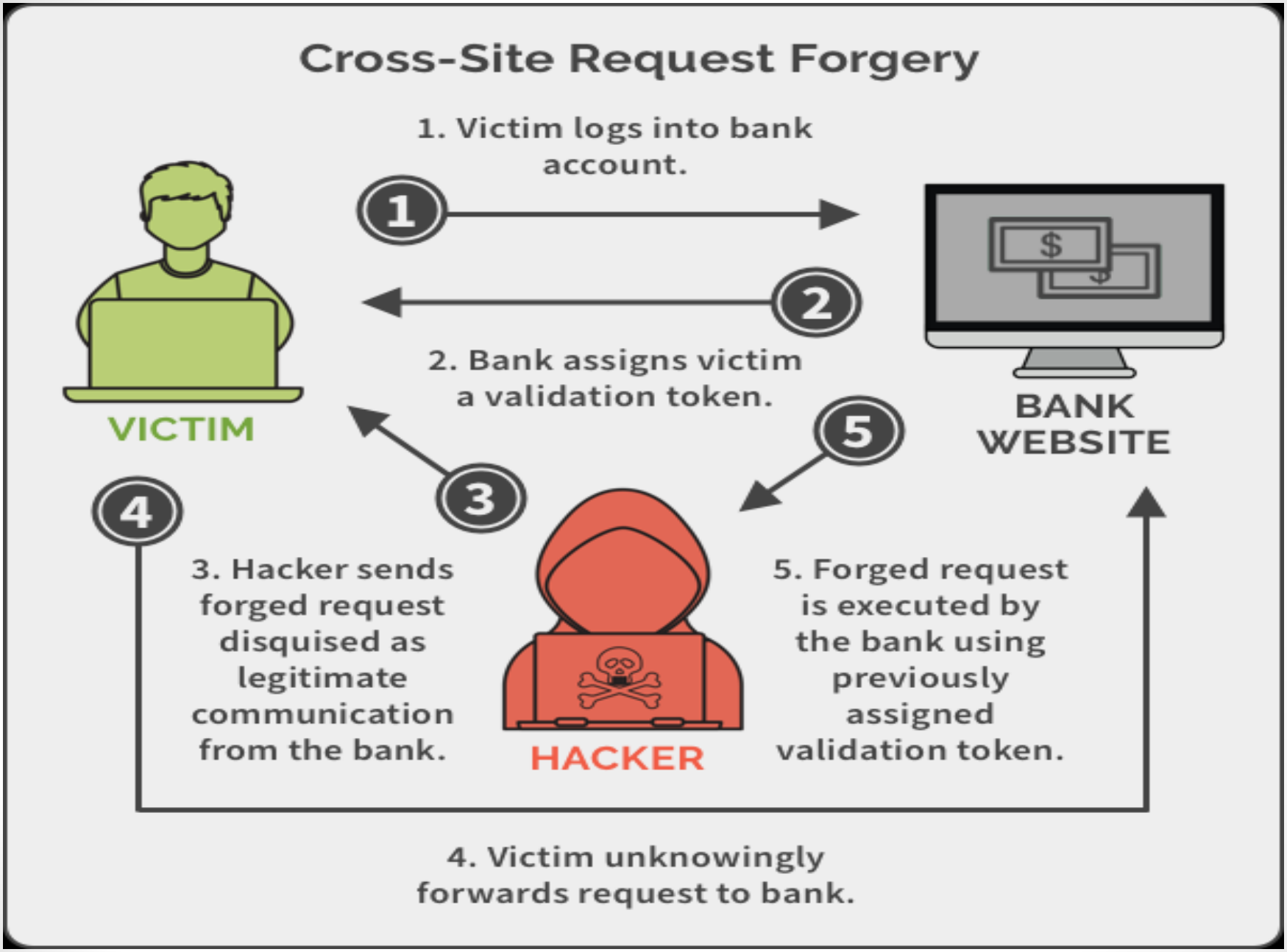
- **Recommendations:**
  - Implement access control checks.
  - Avoid exposing object references in URLs.
  - Verify authorization to all reference objects.

Cross Site Request Forgery(CSRF)

- Cross Site Request Forgery is a forged request came from the cross site.
- CSRF attack is an attack that occurs when a malicious website, email, or program causes a user's browser to perform an unwanted action on a trusted site for which the user is currently authenticated.

- 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.

- A link will be sent by the attacker to the victim when the user clicks on the URL when logged into the original website, the data will be stolen from the website.


Cross-Site Request Forgery
1. Victim logs into bank account.
1
2
2. Bank assigns victim a validation token.
VICTIM
BANK WEBSITE
3
4
5
3. Hacker sends forged request disguised as legitimate communication from the bank.
HACKER
5. Forged request is executed by the bank using previously assigned validation token.
4. Victim unknowingly forwards request to bank.

- **Implication**
  - Using this vulnerability as an attacker can change user profile information, change status, create a new user on admin behalf, etc.
- **Vulnerable Objects**
  - User Profile page
  - User account forms
  - Business transaction page

Example

- The victim is logged into a bank website using valid credentials. He receives mail from an attacker saying "Please click here to donate $1 to cause."
- When the victim clicks on it, a valid request will be created to donate $1 to a particular account.
- **http://www.vulnerablebank.com/transfer.do?account=cause&amount=1**
- The attacker captures this request and creates below request and embeds in a button saying "I Support Cause."
- **http://www.vulnerablebank.com/transfer.do?account=Attacker&amount=1000**
- Since the session is authenticated and the request is coming through the bank website, the server would transfer $1000 dollars to the attacker.
- **Recommendation**
  - Mandate user's presence while performing sensitive actions.
  - Implement mechanisms like CAPTCHA, Re-Authentication, and Unique Request Tokens.

Security Misconfiguration

- Security Configuration must be defined and deployed for the application, frameworks, application server, web server, database server, and platform.
- If these are properly configured, an attacker can have unauthorized access to sensitive data or functionality.

- **implication**
  - Making use of this vulnerability, the attacker can enumerate the underlying technology and application server version information, database information and gain information about the application to mount few more attacks.
- **Vulnerable objects**
  - URL
  - Form Fields
  - Input fields

- The application server admin console is automatically installed and not removed. Default accounts are not changed. The attacker can log in with default passwords and can gain unauthorized access.
- Directory Listing is not disabled on your server. Attacker discovers and can simply list directories to find any file.

Insecure Cryptographic Storage

- Insecure Cryptographic storage is a common vulnerability which exists when the sensitive data is not stored securely.
- The user credentials, profile information, health details, credit card information, etc. come under sensitive data information on a website.
- This data will be stored on the application database. When this data are stored improperly by not using encryption or hashing*, it will be vulnerable to the attackers.
- **Vulnerable objects**
  - Application database.

Failure to restrict URL Access

- **Description**
  - Web applications check URL access rights before rendering protected links and buttons. Applications need to perform similar access control checks each time these pages are accessed.
  - In most of the applications, the privileged pages, locations and resources are not presented to the privileged users.
  - By an intelligent guess, an attacker can access privilege pages. An attacker can access sensitive pages, invoke functions and view confidential information.
- **Implication**
  - Making use of this vulnerability attacker can gain access to the unauthorized URLs, without logging into the application and exploit the vulnerability. An attacker can access sensitive pages, invoke functions and view confidential information.
- **Vulnerable objects:**
  - URLs
- **Examples**
  - Attacker notices the URL indicates the role as "/user/getaccounts." He modifies as "/admin/getaccounts".
  - An attacker can append role to the URL.

Insufficient Transport Layer Protection

- Deals with information exchange between the user (client) and the server (application).
- Applications frequently transmit sensitive information like authentication details, credit card information, and session tokens over a network.

- **Implication**
  - Making use of this web security vulnerability, an attacker can sniff legitimate user's credentials and gaining access to the application.
  - Can steal credit card information.
- **Vulnerable objects**
  - Data sent over the network.
- **Recommendations**
  - Enable secure HTTP and enforce credential transfer over HTTPS only.
  - Ensure your certificate is valid and not expired.

Unvalidated Redirects and Forwards

- **Description**
- The web application uses few methods to redirect and forward users to other pages for an intended purpose.
- If there is no proper validation while redirecting to other pages, attackers can make use of this and can redirect victims to phishing or malware sites, or use forwards to access unauthorized pages.

- An attacker can send a URL to the user that contains a genuine URL appended with encoded malicious URL. A user by just seeing the genuine part of the attacker sent URL can browse it and may become a victim.
- **Examples**
- 1.**http://www.vulnerablesite.com/login.aspx?redirectURL=ownsite.com**
- Modified to
- **http://www.vulnerablesite.com/login.aspx?redirectURL=evilsite.com**