

Name:- Harsh Mittal

Sign:- Harsh

Roll No:- 1022776

Q1.

Ans

Most Common Website Security Vulnerabilities are:-

1. SQL INJECTIONS

SQL injection is a type of web application security vulnerability in which an attacker attempts to use application code to access or corrupt database content. If successful, this allows the attacker to create, read, update, alter, or delete data stored in the back-end database.

&

2. CROSS SITE SCRIPTING (XSS)

Cross site scripting (XSS) targets an application's users by injecting code, usually a client-side script such as JavaScript, into a web application's output. The concept of XSS is to manipulate client-side scripts of a web application to execute in manner desired by the attacker. XSS allows attackers to execute scripts in the victim's browser.

3. BROKEN AUTHENTICATION & SESSION MANAGEMENT

It encompasses several security issues all of them having to do with maintaining the identity of a user. If authentication credentials and session identifiers are not protected at all times, an attacker can hijack an active session and assume the identity of a user.

4. INSECURE DIRECT OBJECT REFERENCES

It is when a web app. exposes a reference to an internal implementation object. Internal implementation objects include files, database records, directories and database keys. When an application exposes a reference to one of these ~~some~~ objects in a URL, Hackers can manipulate to gain access to a user's personal data.

flavours

5. SECURITY MISCONFIGURATION

Security misconfiguration encompasses several types of vulnerabilities all centered on a lack of maintenance or a lack of attention to a web application configuration. A secure configuration must be defined and deployed for the app., framework, application server, web server, database and platform.

6. CROSS-SITE REQUEST FORGERY (CSRF)

It is a malicious attack where a user is tricked into performing an action he or she didn't intend to do. A third-party website will send a request to a web application that a user is already authenticated against.

Harsh