Introduction

World Wide Web has advanced from a framework that delivers static pages to a stage that supports distributed applications, known as web applications and become one of the foremost predominant technologies for information and service delivery over Web.

Web application advancements give a promising system of coordinating numerous useful segments over the web and therefore empower people and associations to cooperate each other utilizing application program interface along enormous topographical separations. Billions of people everywhere throughout the world use web application advancements to exchange data, perform money related exchanges, and have fun and communicate and to socialize themselves [3, 5, 6].Web application grew tremendously in the last few decades and it has brought great benefits to the people, however, these benefits are associated with some challenges and one of the most important challenges is that of security. Security in web application refers to the threat which occurs due to flaws in software design, coding, testing and implementation. Web application services are more prone to cyber attacks due to their public access. And web applications are increasingly used to deliver security critical services; they become a valuable target for security attacks. Many web applications interact with back-end database systems, which may store sensitive information (e.g., financial, health), the compromise of web applications would result in breaching an enormous amount of information, leading to severe economical losses, ethical and legal consequences [7,8]. The Web platform is a complex ecosystem composed of large number of components and technologies, including HTTP protocol, web server and server-side application development technologies (e.g., CGI, PHP, ASP), web browser and client-side technologies (e.g., JavaScript, Flash). Web application built and hosted upon such a complex framework faces characteristic challenges postured by the highlights of those components and innovations and the irregularities among them. Current widely-used web application advancement and testing systems, on the other hand, offer constrained security back. In this way secure web application advancement is an error-prone prepare and requires considerable endeavors, which may be unreasonable beneath time-to-market weight and for individuals with insufficient security skills or awareness. As a result, a high rate of web applications sent on the Web is uncovered to security vulnerabilities. According to a report by the Internet Application Security Consortium, around 49% of the internet applications being looked into contain vulnerabilities of tall hazard level and more than 13% of the websites can be compromised totally naturally [1]. A later report [2] uncovers that over 80% of the websites on the Web have had at least one serious Vulnerability.

Vulnerability refers to a weakness in system’s security requirement, design, coding or operation that could accidently occur or intentionally violated and result in security failure. In last few years, number of reported web application security vulnerabilities has increased. Some commonly reported web application vulnerabilities include SQL injection, cross site scripting, command line injection, cross site request forgery and malicious file execution [3, 4].

**Understanding How A Web Application Works**

Web application could be a distributed application that's executed over the web platform. It is an fundamentally portion of today’s Web biological system that enables dynamic data and benefit conveyance. As shown in Fig. 1, a web application may comprise of code on both the server side and the client side. The server-side code will generate dynamic HTML pages either through execution (e.g., Java servlet, CGI) or elucidation (e.g., PHP,JSP).

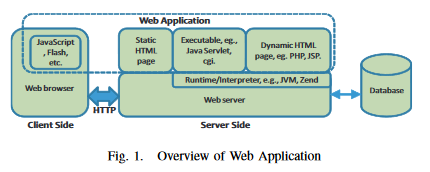


Fig. 1. Overview Of Web Application

During the execution of the server-side code, the internet application may interact with nearby record framework or back-end database for storing and retrieving information. The client-side code(e.g., in JavaScript) are implanted within the HTML pages, which is executed inside the browser. It can communicate with the server-side code (i.e., AJAX) and powerfully overhauls the HTML pages. In what takes after, we portray three one of a kind perspectives of the internet application advancement, which separate web applications from conventional applications.

2.**1 Web Application Architecture**

Web application architecture is often structured as a three-tiered application The architecture of web applications consists of web browser, web server, web application and database server. Tier 1 architecture consists of web browser and web server, Tier 2 for web application and Tier 3 for database.

•Tier 1 – web browser A web browser is also known as web client, functions as the user interface to web server to get input from web application or database server. Web server receives input and interacts with client through web browser by using Hyper Text Transfer Protocol (HTTP) or via secure protocol HTTPS. There are many type of web servers where Apache and Internet Information Server (IIS) are the most popular web server in the world.

•Tier 2 – web application architecture Web application consists of a collection of scripts such as Javascript, VBscript, which reside on a web server and interact with databases or other sources of dynamic content. The common example, the data input using a web browser is processed and stored into database. Java Server Pages (JSP), PHP, Active Server Pages (ASP), Perl and Common Gateway Interface (CGI) are among the technology used to build web based application. Using the infrastructure of the Internet, web applications allow service providers and clients to share and manipulate information in a platform-independent manner. Normally web application server is attached on top of web server and works as interface from web client and database server. Web server will manage the page requested from web client by sending to application server and application server constructs code dynamically and passed back to web server. The flow of data among tiers gives rise to the input validation problem for the web application server; it must check and/or modify incoming input before processing them further or incorporating them into output that it passes to other tiers to execute. Failure to check or sanitize input appropriately can compromise the web application’s security [9].

•Tier 3 – database architecture Stores and manages all the processed users input data. The database tier is responsible for the access of authenticated users and the rejection of malicious users from the database.

Vunlerability Assesment and Penetration Testing:

## Vulnerability Assessment

a vulnerability assessment is the technique of identifying (discovery) and measuring security vulnerabilities (scanning) in a given environment. It is a comprehensive assessment of the information security position (result analysis). Further, it identifies the potential weaknesses and provides the proper mitigation measures (remediation) to either remove those weaknesses or reduce below the risk level.

## Penetration Testing

Penetration testing is the art of finding vulnerabilities and digging deep to find out how much a target can be compromised, in case of a legitimate attack. A penetration test will involve exploiting the network, servers, computers, firewalls, etc., to uncover vulnerabilities and highlight the practical risks involved with the identified vulnerabilities.

## Stages of Penetration Testing

Penetration testing can be broken down into multiple phases; this will vary depending on the organization and the type of test conducted– internal or external. Let’s discuss each phase:

### **1) Agreement phase:**

In this phase, there is a mutual agreement between the parties; the agreement covers high-level details- methods followed and the exploitation levels. The attacker cannot bring down the production server even if the testing has been done at non-peak hours. What if the attacker changes the data that has been contained in the database in production? This will unveil the vulnerabilities but at the cost of business. A non-disclosure agreement has to be signed between the parties before the test starts.

### **2) Planning and reconnaissance:**

In this phase, the attacker gathers as much information about the target as possible. The information can be IP addresses, domain details, mail servers, network topology, etc. An expert hacker will spend most of the time in this phase, this will help with further phases of the attack.

### **3) Scanning:**

This is the phase where the attacker will interact with the target with an aim to identify the vulnerabilities. An attacker will send probes to the target and records the response of the target to various inputs. This phase includes- scanning the network with various scanning tools, identification of open share drives, open FTP portals, services that are running, and much more. In case of a web application, the scanning part can be either dynamic or static. In static scanning, the application code is scanned by either a YTool or an expert application vulnerability analyst. The aim is to identify the vulnerable functions, libraries and logic implemented. In dynamic analysis, the tester will pass various inputs to the application and record the responses; various vulnerabilities like injection, cross-site scripting, remote code execution can be identified in this phase.

### **4) Gaining Access:**

Once the vulnerabilities have been identified, the next step is to exploit the vulnerabilities with an aim to gain access to the target. The target can be a system, firewall, secured zone or server. Be aware that not all vulnerabilities will lead you to this stage. You need to identify the ones that are exploitable enough to provide you with access to the target.

### **5) Maintaining access:**

The next step is to ensure that the access is maintained; i.e., persistence. This is required to ensure that the access is maintained even if the system is rebooted, reset or modified. This kind of persistence is used by attackers who live in the system and gain knowledge about them over a period of time, and when the environment is suitable, they exploit.

### **6) Exploitation:**

This is the phase where the actual damage is done. An attacker will try to get the data, compromise the system, launch dos attacks, etc. Usually, this phase is controlled in penetration testing so as to ensure that the mayhem on the network is limited. This phase is modified in this way- a dummy flag is placed in the critical zone, may be in the database; the aim of the exploitation phase will be to get the flag. Revealing the contents of the flag will be enough to ensure practical exploitation of the network or data theft.

### **7) Evidence collection and report generation:**

Once the penetration test is complete, the final aim is to collect the evidence of the exploited vulnerabilities and report it to the executive management for review and action. Now, it is the management’s decision on how this risk has to be addressed. Whether they want to accept the risk, transfer it or ignore it (least likely option).

## Different Types and Methods of Penetration Testing

[Types of penetration testing](https://www.greycampus.com/opencampus/ethical-hacking/penetration-testing) can be categorized on the basis of either, the knowledge of the target or the position of the penetration tester. There are a few other parameters to the categorization of penetration.

### **Black Box, Gray Box, and White Box:**

When the penetration tester is given the complete knowledge of the target, this is called a white box penetration test. The attacker has complete knowledge of the IP addresses, controls in place, code samples, etc. When the attacker has no knowledge of the target, this is referred to as a black box penetration test. Please note that the tester can still have all the information that is publically available about the target. When the tester is having partial information about the target, this is referred to as gray box penetration testing. In this case, the attacker is having some knowledge of the target like URLs, IP addresses, etc., but does not have complete knowledge or access. This is with respect to the knowledge.

### **Internal and External Penetration test:**

If the penetration test is conducted from outside the network, this is referred to as external penetration testing. If the attacker is present inside the network, simulation of this scenario is referred to as internal penetration testing. Since the attacker is an internal person, the knowledge about the system and the target will be abundant when compared to a test conducted from outside.

### **In-house and Third party Penetration test:**

When the test is conducted by an in-house security team, it is another form of internal penetration testing. Companies often hire third-party organizations to conduct these tests, this is referred to as third-party penetration testing.

### **Blind and Double-Blind Penetration test:**

In a blind penetration test, the penetration tester is provided with no prior information but the organization name. The penetration tester will have to do all the homework, just like a legitimate attacker would do. This will surely take more time, but the results would be more close to the practical attacks. A double-blind test is like a blind test but the security professionals will not know when the testing will start. Only the senior management will have this information. This will test the processes, controls and the awareness of the security teams if and when a real attack occurs.

Web application Pen Testing Tools :

### **1. Nessus**

Nessus is a network and web application [vulnerability scanner](https://www.greycampus.com/opencampus/ethical-hacking/hacking-methodology), it can perform different types of scans and help a penetration tester identify vulnerabilities. The attacker can then spend time in determining what can be exploited further. The free version of the tool is having some interesting features disabled. The full version is powerful and has a lot of features that will help during the scanning phase of the penetration test.

### **2. Dirbuster**

Dirbuster is a directory busting tool, this will help the attacker to find the directories that are present. The tool will take an input list and will help in testing their availability. This will allow for footprinting of the directory structure and find directories that will be difficult to find.

### **3. Metasploit**

Metasploit is an exploitation framework that has been packed with various capabilities. A skilled attacker can generate payloads, shellcodes, gain access, and perform privilege escalation attacks. The knowledge of python and ruby will be helpful since the framework uses them for most of the scripts.

### **4. Burp Suite**

This tool is specifically used for testing web applications. Let us assume that you have uncovered a test web application that is no longer used after production push. You can use this tool to dig deeper into the application and hunt vulnerabilities. The high severity vulnerabilities can be further exploited to move forward with the attack.

**Web application vulnerabilities**

Because web applications are open to the world, they are more vulnerable to attacks and prone to a great variety of vulnerabilities. In this section, we describe some of the most common and well-known web application vulnerabilities based on OWASP Top Ten lists 2010 [10]. OWASP stands for Open Web Applications Security Project, and is an open-source collaboration of web based security tools, technologies and methodologies from industry leaders, educational organizations and individuals from around the world. The OWASP Top Ten is a valuable document for developers and testers because its focus on web applications. The OWASP Top 10 2017 has listed the ten most critical web application security vulnerabilities as shown in Table 1. The OWASP Top 10 2017 refers to the top 10 web attacks as seen over the year by security experts, and community contributors to the project.

|  |
| --- |
| OWASP TOP 10 |
| A1:2017-injection |
| A2:2017-Broken Authentication |
| A3:2017-Sensitive Data Exposure |
| A4:2017-Xml External Entities(XXE) |
| A5:2017-Broken Access control |
| A6:2017-Security misconfiguration |
| A7:2017-Cross-site Scripting |
| A8:2017-Insecure Deserialization |
| A9:2017-Using components with known Vulnerabilities |
| A10:2017-Insufficient logging and Monitoring |

**Table1.** OWASP Top Ten Vulnerability 2017 [10]

**Injection:**

Injection flaws, such as SQL, NoSQL, OS, 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. Almost any source of data can be an injection vector, environment variables, parameters, external and internal web services, and all types of users. Injection flaws occur when an attacker can send hostile data to an interpreter[12].

Injection flaws are very prevalent, particularly in legacy code. Injection vulnerabilities are often found in SQL, LDAP, XPath, or NoSQL queries, OS commands, XML parsers, SMTP headers, expression languages, and ORM queries. Injection flaws are easy to discover when examining code. Scanners and fuzzers can help attackers find injection flaws. Injection can result in data loss, corruption, or disclosure to unauthorized parties, loss of accountability, or denial of access[12]. Injection can sometimes lead to complete host takeover. The business impact depends on the needs of the application and data.

An application is vulnerable to attack when:

1. User-supplied data is not validated, filtered, or sanitized by the application.
2. Dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter.
3. Hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records…
4. Hostile data is directly used or concatenated, such that the SQL or command contains both structure and hostile data in dynamic queries, commands, or stored procedures.

The concept is identical among all interpreters. Source code review is the best method of detecting if applications are vulnerable to injections, closely followed by thorough automated testing of all parameters, headers, URL, cookies, JSON, SOAP, and XML data inputs. Organizations can include static source (SAST) and dynamic application test (DAST) tools into the CI/CD pipeline to identify newly introduced injection flaws prior to production deployment.

The most common injection types are:

* Expression Language (EL)
* Object Graph Navigation Library (OGNL)
* LDAP
* Object Relational Mapping (ORM)
* OS command
* NoSQL
* SQL

SQL is possibly the most common and widespread injection type. Instead of a Python, Pearl, or other script type, malicious SQL commands are embedded into the content of the user input [11].

**Sql injection:**

SQL injection is a web security vulnerability that allows an attacker to interfere with the queries that an application makes to its database. It generally allows an attacker to view data that they are not normally able to retrieve. This might include data belonging to other users, or any other data that the application itself is able to access. In many cases, an attacker can modify or delete this data, causing persistent changes to the application's content or behavior In some situations, an attacker can escalate an SQL injection attack to compromise the underlying server or other back-end infrastructure, or perform a denial-of-service attack.

These common mistakes must be avoided with good programming habits. The programmer must apply the following methods for the protection from SQLIA's:-

* Avoid building dynamic SQL statement from user input.
* Length of input string must be limited.
* Escape query delimiter, SQL keyword, character data string delimiter and single line comment in user input.
* Use different Database account for different level of privileges.
* Error messages must be customized to hide the details of injectable parameters.
* Use parameterized queries for Database access.
* Use stored procedures to avoid direct access of Database.
* Void building SQL statements from cookie and HTTP variables.

Impact of SQL Injection As we already mentioned SQL injection attack is accomplished by providing data (inclusion of SQL queries) from an external source which is further used to dynamically construct a SQL query. The impact and consequences of SQL injection attacks can be classified as follows:

1. **Confidentiality:** Confidentiality loss is a significant issue with SQL Injection attacks since SQL databases generally hold sensitive and critical information which could be viewed by unauthorized users as a consequence of successful SQL injection attack.
2. **Integrity:** Successful SQL injection attack allows external source to make unauthorized modifications such as changing or even deleting destination database data.
3. **Authentication:** Poorly formulated SQL queries do not correctly validate User identity that allow unauthenticated users act as an authenticated user without out knowledge of the password or even user name.
4. **Authorization:** The successful exploitation of SQL injection vulnerability enables an attacker to alter permission and obtain higher privileges ,data in the database. It is very difficult to identify SQL injecting before effect in reality.In most number of scenarios, The unlawful transaction is carried out by the intruder using legitimate customer identification or the intrinsic characteristics of the database application such as malicious modification of existing SQL Queries of web application that are accessing critical sections of the affected databases.

**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 [10].

Broken Authentication is a kind of web vulnerability which occurs due to the misconfiguration of session management. After an authentication process completed, a session will be created which will be activated for data communication between the server and a particular user. Fig. 1 represents the problem of Broken Authentication by exploiting session mismanagement problem. If any intruder can get access in the active session of any specific user bypassing the authentication process, the scenario is treated as broken Exploiting Authentication problem of the given application. Fig. 1 represents the overall process of user authentication and session management.

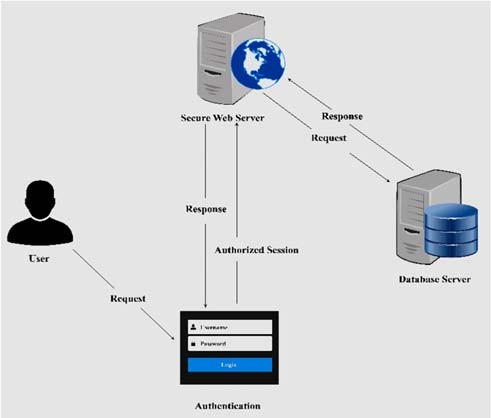


Figure 1. Authentication and Session Management Process

A session request is raised by a user of a web application through the login page where the user credential has been provided. Once the given request has been sent from the client side to server side, the server initiates a query to the database for checking whether the user provided credential is matched [13].

A session request is raised by a user of a web application through the login page where the user credential has been provided. Once the given request has been sent from the client side to server side, the server initiates a query to the database for checking whether the user provided credential is matched with the record of the database or not. As soon as the validation process is successful, a session with a specific ID will be allocated for the user to communicate the application. A user then can access the system with a given privileges provided by the administrator of the system for getting different services.

There is some exploitation techniques used to exploit Broken Authentication & Session Management. The Types are given below:

1. *Session Misconfiguration Attack:*

Session duration isone of the major facts in maintaining a secure authentication process of the web applications. As soon as the user credential is validated from a system, it assigns a session for the particular user with a session ID for a limited period of time. In case the developer of the web application sets the session duration parameter with a large value, the session will remain active for that specific period if the user not logged off their account as directed by the designer of the application. Therefore, that session can be reestablished to re-using by an intruder which leads to Broken Authentication. Session misconfiguration is one of the most critical areas for Broken Authentication and Session Management vulnerability.

1. *Using Cracking/ Guessing Weak Password Exploitation*

Due to lack of awareness about password management, some non-technical users keep their password in a generalize form like admin, password, my password, password123, admin1997 etc. and also in some cases, user remains the default password for their access into the system which will be easy to guess for an attacker to get access in the system. It is an automated process of cracking/ guessing user’s weak passwords. Attacker gives user login link in Hydra in which it checks predefined dataset for trying to find username and password

1. *Exploiting Authentication problem*

Web applications authentication systems are handled by using conditional quires to check username and password against one user for authentication. If these conditional queries get infected or not properly handled, it could easily compromised by an intruder to get access into the system without proper authentication.

1. *Decoding Inadequate Encryption*

In some web applications privacy measures are not properly handled by the developers. Therefore, an attacker can steal the session ID against one user by exploiting the security flaws of disclosing the session ID in the URL of the system,

*e.g. http://www.demosite.com/transactions/saleitems?sessioni d=7892384838&dest=demo user*

The example shows the general transaction’s session id of demo user has been disclosed publicly in the URL. As such, it is not very critical for an attacker to steal some other user’s session id just only changing the session ID value into the URL. The attack process is feasible for the inadequate encryption in the value of session ID. After changing the value in session ID, it will look like as below: *http://www.demosite.com/transactions/saleitems?sessionid= 7892384839&dest=attacker name.*

1. *Other Vulnerabilities:*

Web application vulnerabilities allow users to disclose users/ systems sensitive information. It also causes major harm to other circumstance e.g. it allows users to execute malicious quires in the system if the system is vulnerable to XSS vulnerability, it also allows attackers to post malicious links for phishing to steal session of the victim, etc. Forgotten password functionally, relying on IP address for session, emailing user credentials, not authenticating a user before changing password, and not having adequate timeouts for inactive session are also reason for Broken Authentication.

**How to Prevent**

* Where possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
* Do not ship or deploy with any default credentials, particularly for admin users.
* Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.
* Align password length, complexity and rotation policies with NIST 800-63 B's guidelines in section 5.1.1 for Memorized Secrets or other modern, and evidence based password policies [14].
* Ensure registration, credential recovery, and API pathways are hardened against account enumeration attacks by using the same messages for all outcomes.
* Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
* Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session IDs should not be in the URL, be securely stored and invalidated after logout, idle, and absolute timeouts.

**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 may be compromised without extra protection, such as encryption at rest or in transit, and requires special precautions when exchanged with the browser [10].

The potential impact is always considered high. What the data consists of varies and so does the impact. The danger lies in the data being exposed, and the potential impact reflects the data’s sensitivity. For example, if credit card data is stolen, the attacker can empty the victim’s bank account. If passwords are exposed, the attacker can abuse these credentials. If certificates are stolen, the attacker can pretend to be the target. It all depends on what kind of data is at risk of being exposed.

The first thing is to determine the protection needs of data in transit and at rest. For example, passwords, credit card numbers, health records, personal information and business secrets require extra protection, particularly if that data falls under privacy laws, e.g. EU's General Data Protection Regulation (GDPR), or regulations, e.g. financial data protection such as PCI Data Security Standard (PCI DSS) [12]. For all such data:

* Is any data transmitted in clear text? This concerns protocols such as HTTP, SMTP, and FTP. External internet traffic is especially dangerous. Verify all internal traffic e.g. between load balancers, web servers, or back-end systems.
* Is sensitive data stored in clear text, including backups?
* Are any old or weak cryptographic algorithms used either by default or in older code?
* Are default crypto keys in use, weak crypto keys generated or re-used, or is proper key management or rotation missing?
* Is encryption not enforced, e.g. are any user agent (browser) security directives or headers missing?
* Does the user agent (e.g. app, mail client) not verify if the received server certificate is valid? [12].

**To prevent the sensitive data exposure:**

* Classify data processed, stored, or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
* Apply controls as per the classification.
* Don’t store sensitive data unnecessarily. Discard it as soon as possible or use PCI DSS compliant tokenization or even truncation. Data that is not retained cannot be stolen.
* Make sure to encrypt all sensitive data at rest.
* Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.
* Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security (HSTS).
* Disable caching for responses that contain sensitive data.
* Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt, or PBKDF2.
* Verify independently the effectiveness of configuration and settings[12]..

**Xml External Entities (XXE)**

Many older or poorly configured XML processors evaluate external entity references within XML documents. External entities can be used to disclose internal files using the file URI handler, internal file shares, internal port scanning, remote code execution, and denial of service attacks[10].

**Broken Access control**

Restrictions on what authenticated users are allowed to do are often 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[10].

**Security misconfiguration**

Security misconfiguration is the most commonly seen issue. This is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information. Not only must all operating systems, frameworks, libraries, and applications be securely configured, but they must be patched and upgraded in a timely fashion[10].

**Cross-site Scripting**

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 HTML or 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[10].

**Insecure Deserialization**

Insecure Deserialization often leads to remote code execution. Even if Deserialization flaws do not result in remote code execution, they can be used to perform attacks, including replay attacks, injection attacks, and privilege escalation attacks[10].

**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 defenses and enable various attacks and impacts [10].

**Insufficient logging and Monitoring**

Insufficient logging and monitoring, coupled with missing or ineffective integration with incident response, allows attackers to further attack systems, maintain persistence, pivot to more systems, and tamper, extract, or destroy data. Most breach studies show time to detect a breach is over 200 days, typically detected by external parties rather than internal processes or monitoring [10].