Experiment 2

Name	Ameya S. Daddikar
College I.D.	161070015
Course	Btech. Computer Engineering

Aim

To study and implement OWASP attacks.

Theory

Injection

Injection flaws, such as SQL, QS, and LDAP injection occur when untrusted data is sent to an interpreter as part of a command or query. The attacker's hostile data ran trick the interpreter into executing unintended commands or accessing data without proper Authorization.

Mitigation

- SQL injection filtering works in a similar way to email's spam filters. Database firewalls detect SQL injections based on the number of invalid queries from host, while there are OR and UNION blocks inside of request, or others.
- With most development platforms, parameterized statements that work with parameters can be used(sometimes called placeholders or bind variables) instead of embedding user input in the statement. A placeholder can only store a value of the given type and not an arbitrary SQL fragments. Hence the SQL injection would simply be treated as a strange (and probably invalid) parameter value.

Broken Authentication and Session Management

Application functions related to authentication and session management are often not implemented correctly, allowing attackers to compromise passwords, keys, or session tokens, or to exploit other implementation flaws to assume other users' identities.

Mitigation

- **Password Strength** - passwords should have restrictions that require a minimum size and complexity for the password.
- **Password Storage** - All passwords must be stored in either hashed or encrypted form to protect them from exposure, regardless of where they are stored.

Cross-Site Scripting (XSS)

XSS flaws occur whenever an application takes untrusted data and sends it to a web browser without proper validation or escaping. XSS allows attackers to execute scripts in the victim's browser, which can hijack user sessions, deface websites, or redirect the user to malicious sites.

Mitigation

- Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, ReactJS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered.
- Escaping untrusted HTTP request data based on the context in the HTML output.

Broken Access control

Broken access control occurs if a user is able to access unauthorized resources, this can be
access to restricted pages, database, directories et cetera. Applications have various account types depending on the users: admins, operators and reporting groups etc. One common problem is that the developers restrict the privileges just on the UI side and not on the server side. If exploited, each user can have admin rights.

Mitigation

- Disable Client Side Caching – Many users access web applications from shared computers located in libraries, schools, airports, and other public access points. Browsers frequently cache web pages that can be accessed by attackers to gain access to otherwise inaccessible parts of sites. Developers should use multiple mechanisms, including HTTP headers and meta tags, to be sure that pages containing sensitive information are not cached by user's browsers.
- Forced Browsing Past Access Control Checks – many sites require users to pass certain checks before being granted access to certain URLs that are typically 'deeper' down in the site. These checks must not be bypassable by a user that simply skips over the page with the security check.

Security Misconfiguration

Good security requires having a secure configuration defined and deployed for the application, frameworks, application server, web server, database server, and platform. Secure settings should be defined, implemented, and maintained, as defaults are often insecure. Additionally, software should be kept up to date.

Mitigation

- A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.

- A segmented application architecture that provides effective, secure separation between components or tenants, with segmentation, containerization, or cloud security groups (ACLs).

Sensitive Data Exposure

Many web applications do not properly protect sensitive data, such as credit cards, tax IDs, and authentication credentials. Attackers may steal or modify such weakly protected data to conduct credit card fraud, identity theft, or other crimes. Sensitive data deserves extra protection such as encryption at rest or in transit, as well as special precautions when exchanged with the browser.

Mitigation

- Classify data processed, stored or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs.
- Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.

XML External Entity

An XML External Entity attack is a type of attack against an application that parses XML input. This attack occurs when XML input containing a reference to an external entity is processed by a weakly configured XML parser. This attack may lead to the disclosure of confidential data, denial of service, server side request forgery, port scanning from the perspective of the machine where the parser is located, and other system impacts.

Mitigation

- Since the whole XML document is communicated from an untrusted client, it's not usually possible to selectively validate or escape tainted data within the system identifier in the DTD. Therefore, the XML processor should be configured to use a local static DTD and disallow any declared DTD included in the XML document.

Cross-Site Request Forgery (CSRF)

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. This allows the attacker to force the victim's browser to generate requests the vulnerable application things are legitimate requests from the victim.

Mitigation

- The preferred option is to include the unique token in a hidden field. This causes the value to be sent in the body of the HTTP request, avoiding its inclusion in the URL, which is more prone to exposure.
- Requiring the user to reauthenticate, or prove they are a user (e.g., via a CAPTCHA) can also protect against CSRF.

Using Components with Known Vulnerabilities

Components, such as libraries, frameworks and other software modules, almost run with full privileges. If a vulnerable component is exploited, such an attack can facilitate serious data loss or server takeover. Applications using components with known vulnerabilities may undermine application defenses and enable a range of possible attacks and impacts.

Mitigation

- Remove unused dependencies, unnecessary features, components, files, and documentation.
- Only obtain components from official sources over secure links. Prefer signed packages to reduce the chance of including a modified, malicious component.

Insufficient logging and monitoring

With all the countermeasures in place attacks still happen and that gets noticed only after an incident has happened. If undetected the attackers could have compromised the systems long back and gained persistence. To ensure the malicious intent of the attackers gets noticed beforehand, it is essential to log all the activity and monitor it for any suspicious behavior.

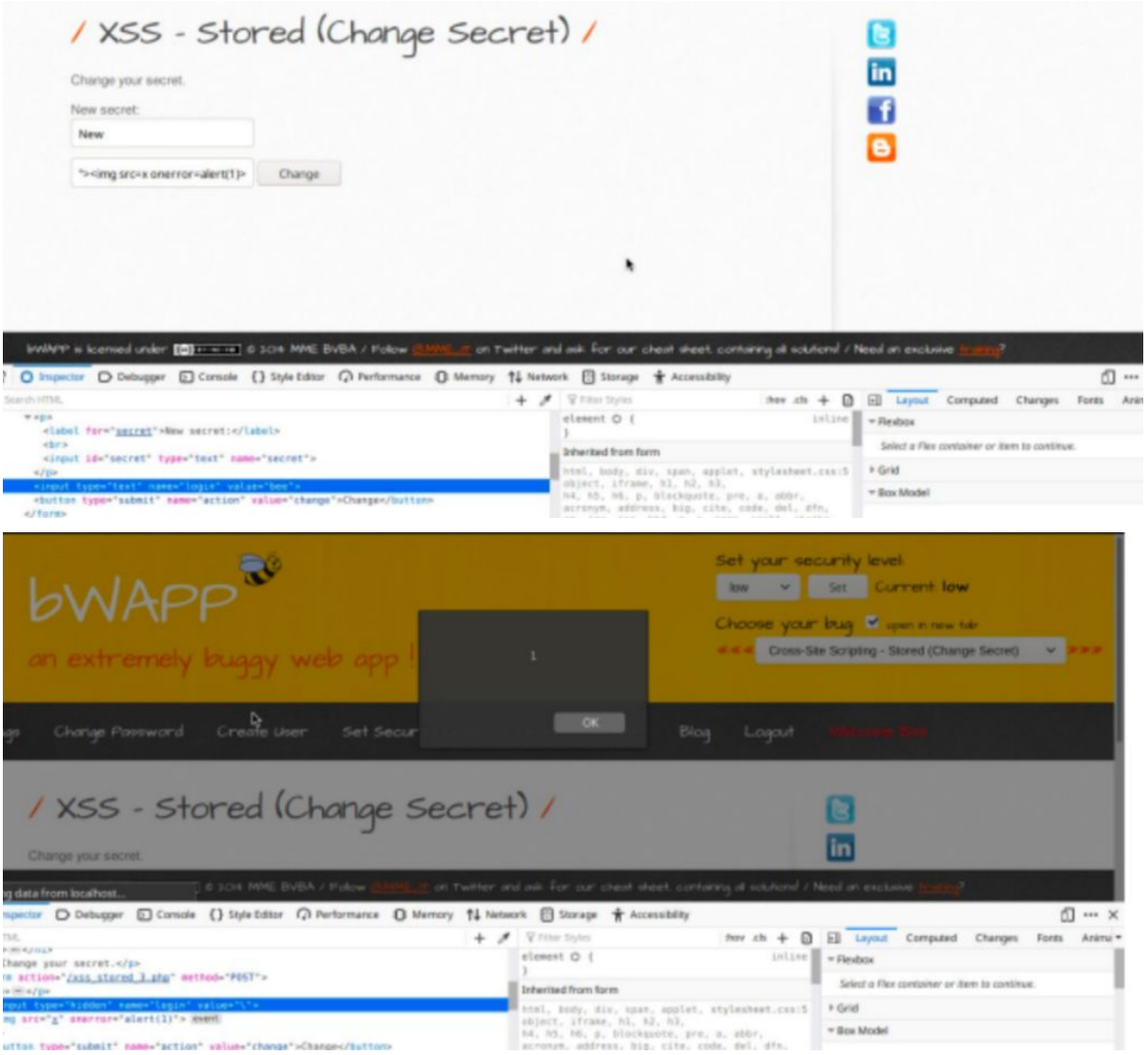
Mitigation

- Ensure that logs are generated in a format that can be easily consumed by a centralized log management solutions.
- Establish effective monitoring and alerting such that suspicious activities are detected and responded to in a timely fashion.

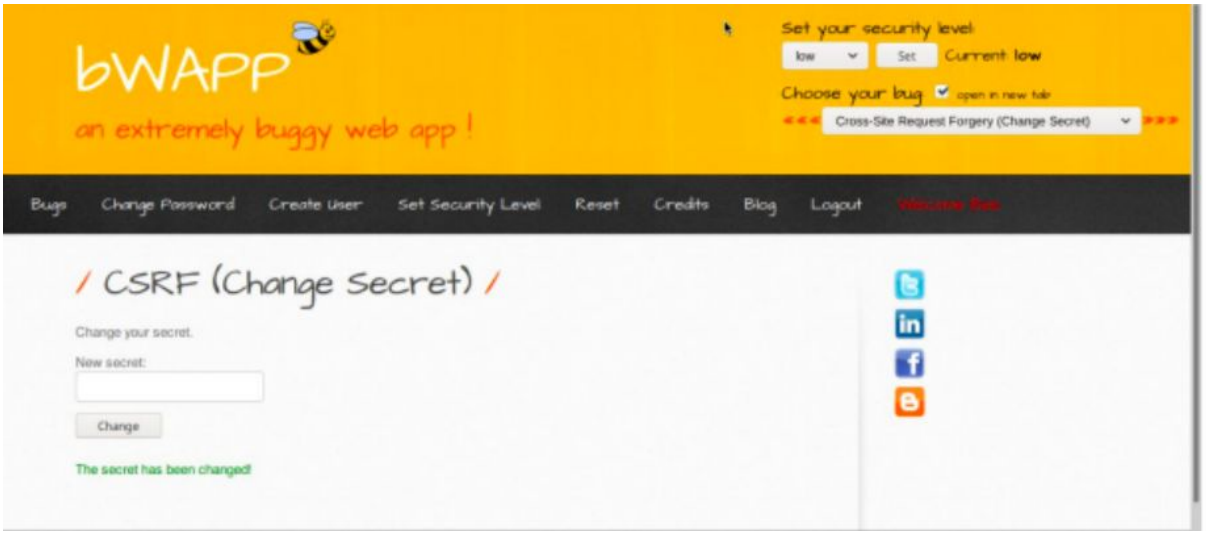
Outputs

SQL Injection

XSS Attack



CSRF Attack



Conclusion

Thus we have studied and replicated OWASP attacks in a controlled and monitored environment to get a better understanding of their working and possible mitigation strategies.