Technical Security Report

1. Special Notes

This report has been generated for the sole purpose of penetrating http://demo.testfire.net/ and its sub-domains. Any use of this document other than security management is strictly prohibited.

2. Legal Disclaimer

This report and any supplements are **CONFIDENTIAL** and may be protected by one or more legal privileges. It is intended solely for the use of the addressee identified in the report. If you are not the intended recipient, any use, disclosure, copying or distribution of the report is **UNAUTHORIZED**. If you have received this report in error, please destroy it immediately.

By accepting this report, you understand that assessing computer security is highly complex and changeable. I Mr. Sanil Almeida (CEH | CPTE) make no warranty that I will find every vulnerability in your Web Application or Web Server(s), or that the solutions suggested and advice provided in this report will be complete or error-free. I shall be held harmless and free from all liabilities for any use or application of the information provided by http://demo.testfire.net/ in connection with using the "Services".

3. Technical Summary

3.1. Scan Session Information

URL :	http://demo.testfire.net/
Date:	November 8th, 2020
Scan Policy:	OWASP Policy
SSL Cipher (Algorithm):	N/A
Server Reported:	Apache-Coyote/1.1 PHP/5.3
Server-side Technologies:	[PHP]

3.2. Issues Found

4.1 SQL Injection……04

4.1.1 False positives for SQLi……07

4.2 Cross Site Scripting (XSS)……09

4.3 Path Traversal……13

4.4 Information Disclosure……16

4.5 References & Useful Links……19

4. Items that require your attention

4.1. SQL Injection

What is SQL injection (SQLi)?

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 behaviour.

What is the impact of SQL injection attack?

Effective SQL injection attacks can result in unauthorized access to sensitive data, such as passwords , credit card information or personal user data. In recent years, several high-profile data breaches seem to be the result of SQL injection attacks, leading to reputation loss and regulatory fines.

How to prevent it.

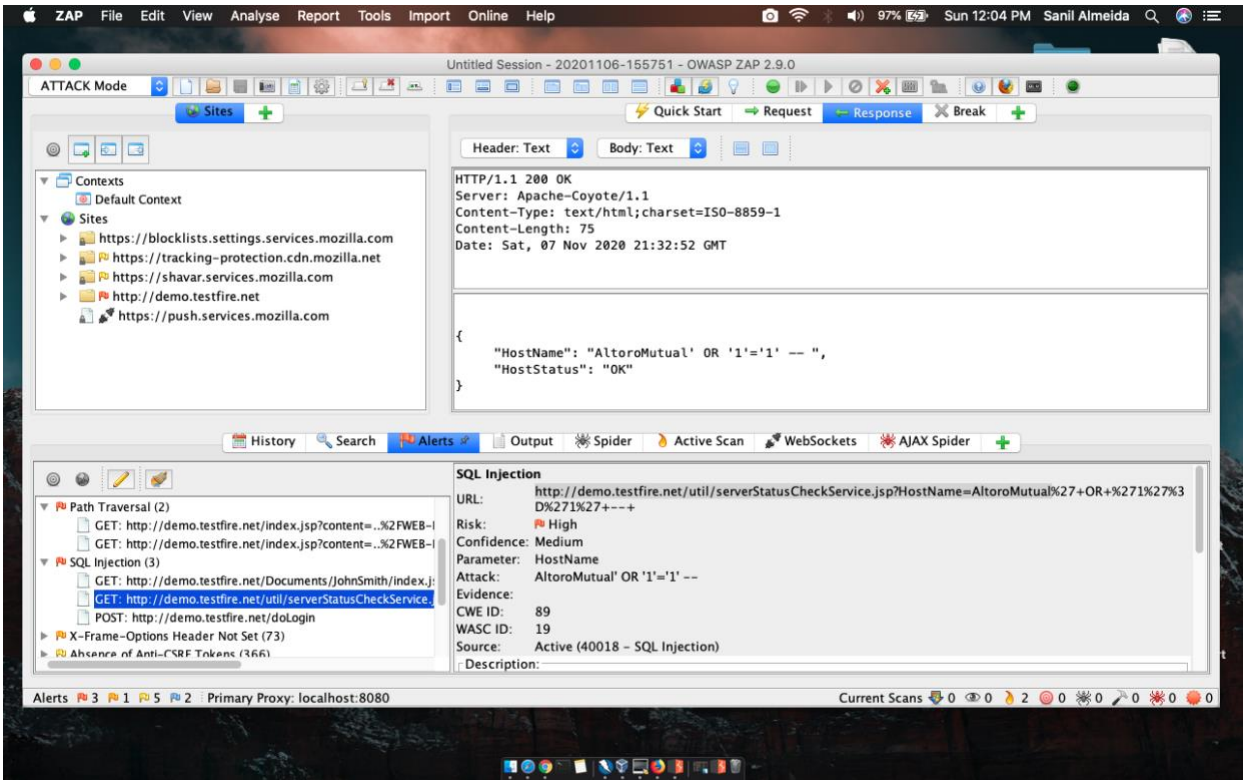
1. Do not trust client side input, even if there is client side validation in place.
2. In general, type check all data on the server side.
3. If the application uses JDBC, use PreparedStatement or CallableStatement, with parameters passed by '?'
4. If the application uses ASP, use ADO Command Objects with strong type checking and parameterized queries.
5. If database Stored Procedures can be used, use them.
6. Do *not* concatenate strings into queries in the stored procedure, or use 'exec', 'exec immediate', or equivalent functionality!
7. Do not create dynamic SQL queries using simple string concatenation.
8. Escape all data received from the client.
9. Apply a 'whitelist' of allowed characters, or a 'blacklist' of disallowed characters in user input.
10. Apply the principle of least privilege by using the least privileged database user possible.

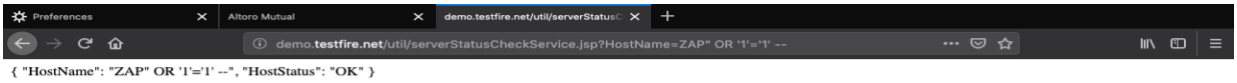
11. In particular, avoid using the 'sa' or 'db-owner' database users. This does not eliminate SQL injection, but minimizes its impact.
12. Grant the minimum database access that is necessary for the application.

List of vulnerable links to SQLi attack.

1. Attack 1

- Vulnerable URL: http://demo.testfire.net/util/serverStatusCheckService.jsp?HostName=AltoroMutual' OR '1'='1'--
- Risk: High
- Injection Parameter: HostName
- Attack: AltoroMutual' OR '1'='1'--
- Proof of Concept:





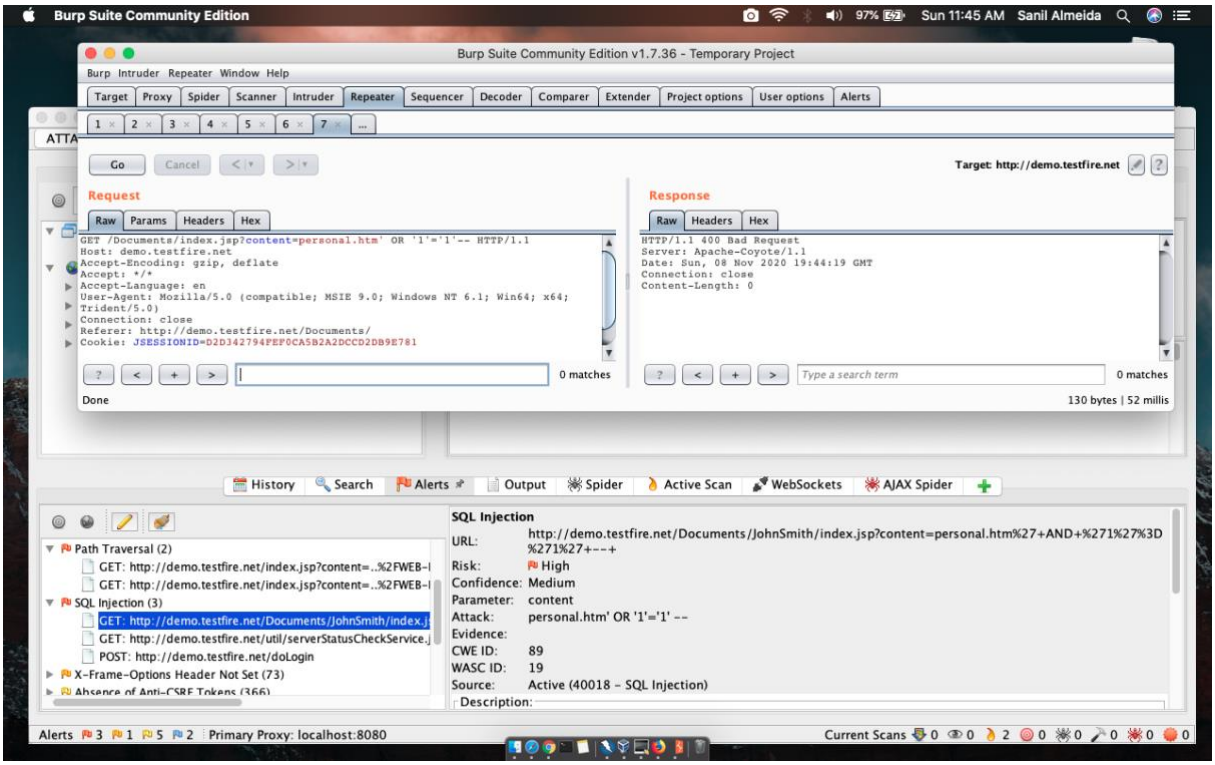
False Positives

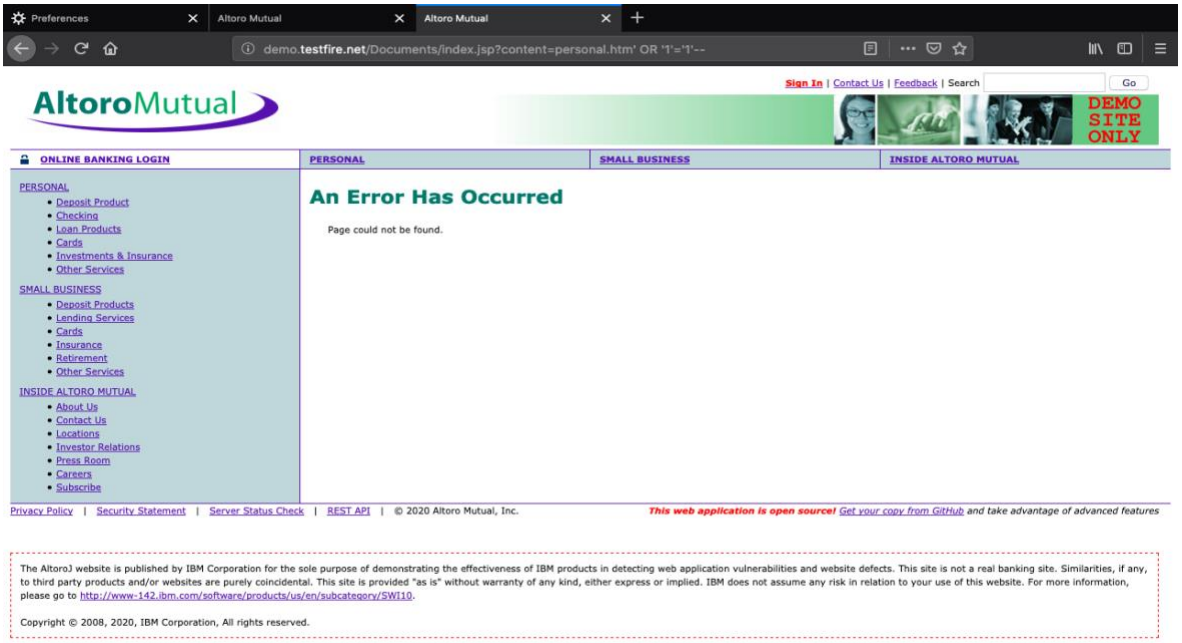
False Positives occur when a scanner, Web Application Firewall (WAF), or Intrusion Prevention System (IPS) flags a security vulnerability that you do not have. These false/non-malicious alerts (SIEM events) increase noise and can include software bugs, poorly written software, or unrecognized network traffic.

4.1.1 False Positive vulnerable links to SQLi attack.

1. Attack 1

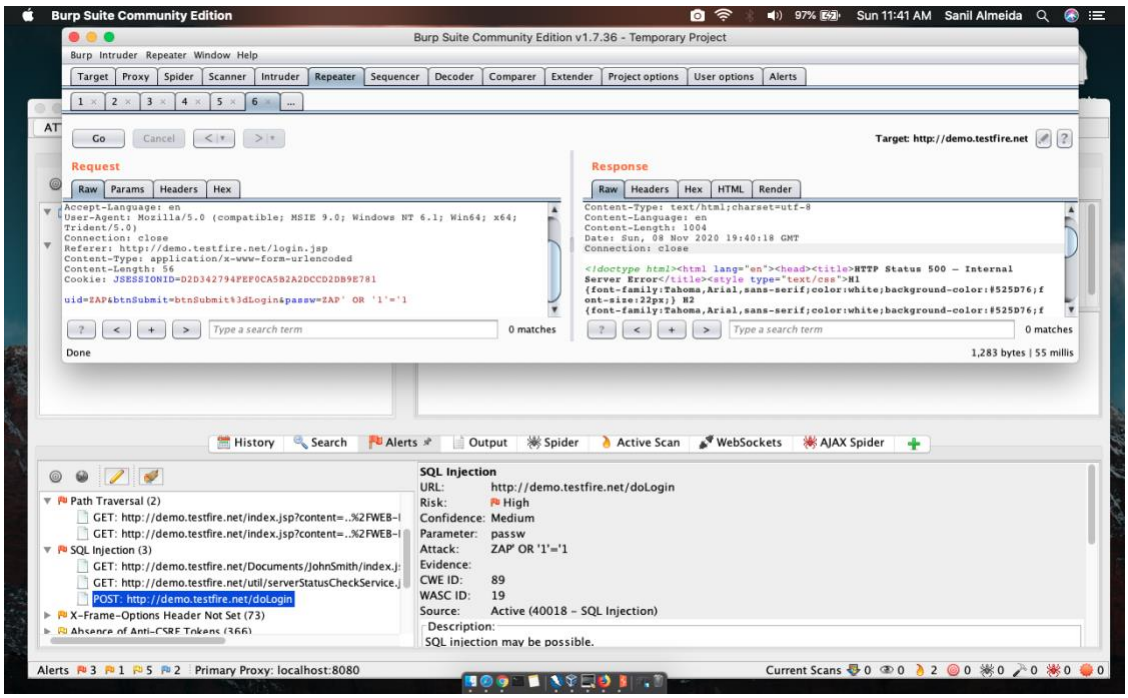
- Vulnerable URL: http://demo.testfire.net/Documents/index.jsp?content=personal.htm' OR '1'='1'--
- Risk: High
- Injection Parameter: content
- Attack: personal.htm' OR '1'='1'--
- Proof of Concept:

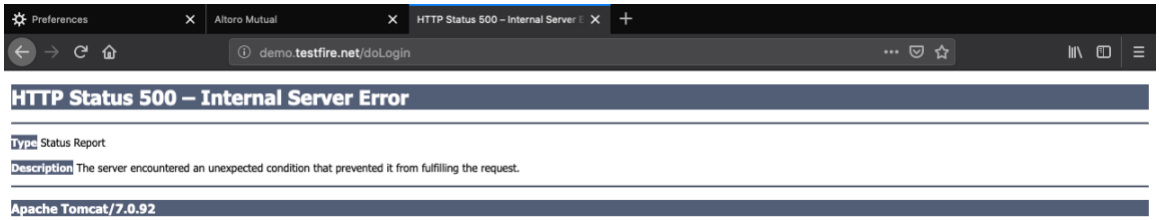




2. Attack 2

- Vulnerable URL: http://demo.testfire.net/doLogin
- Risk: High
- Injection Parameter: passw
- Attack: ZAP OR '1'='1
- Proof of Concept:





4.2. Cross Site Scripting (XSS)

What is Cross Site Scripting (XSS)?

Cross-Site Scripting (XSS) attacks are a type of injection, in which malicious scripts are injected into otherwise benign and trusted websites. XSS attacks occur when an attacker uses a web application to send malicious code, generally in the form of a browser side script, to a different end user. Flaws that allow these attacks to succeed are quite widespread and occur anywhere a web application uses input from a user within the output it generates without validating or encoding it.

An attacker can use XSS to send a malicious script to an unsuspecting user. The end user's browser has no way to know that the script should not be trusted, and will execute the script. Because it thinks the script came from a trusted source, the malicious script can access any cookies, session tokens, or other sensitive information retained by the browser and used with that site. These scripts can even rewrite the content of the HTML page.

What is the impact of XSS attack?

The impact of cross-site scripting vulnerabilities can vary from one web application to another. It ranges from session hijacking to credential theft and other security vulnerabilities. By exploiting a cross-site scripting vulnerability, an attacker can impersonate a legitimate user and take over their account.

If the victim user has administrative privileges, it might lead to severe damage such as modifications in code or databases to further weaken the security of the web application, depending on the rights of the account and the web application.

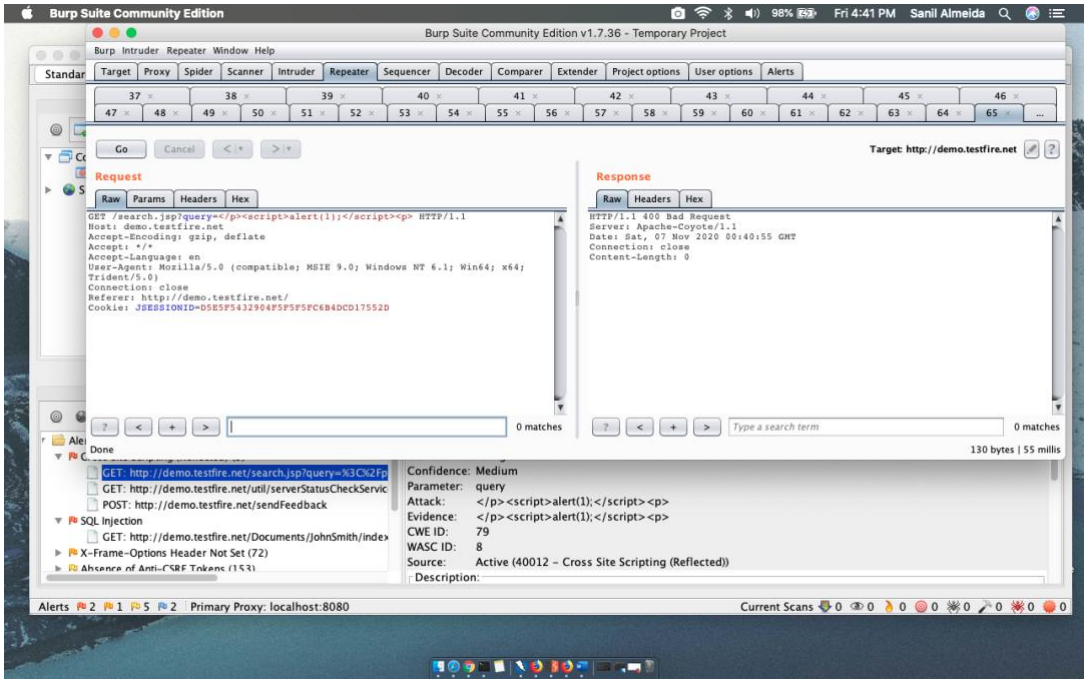
How to prevent it.

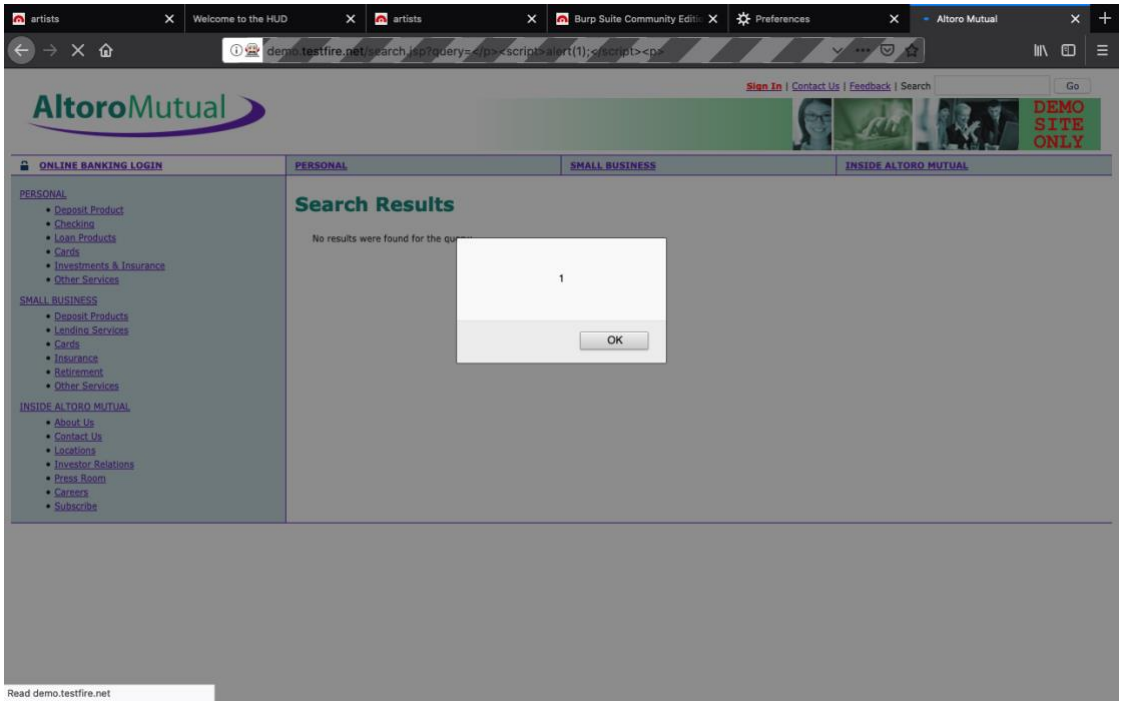
1. Sanitize data input in an HTTP request before reflecting it back, ensuring all data is validated, filtered or escaped before echoing anything back to the user, such as the values of query parameters during searches.
2. Convert special characters such as ?, &, /, <, > and spaces to their respective HTML or URL encoded equivalents.
3. Protect your web application from various forms of cross-site scripting by using HTML entity encoding before sending untrusted data into a browser.
4. Use security headers and set the HTTPOnly flag of your session cookie and other custom cookies you may have that are not accessed by any JavaScript codes you wrote.
5. Implement content security policy.

List of vulnerable links to XSS attack.

1. Attack 1

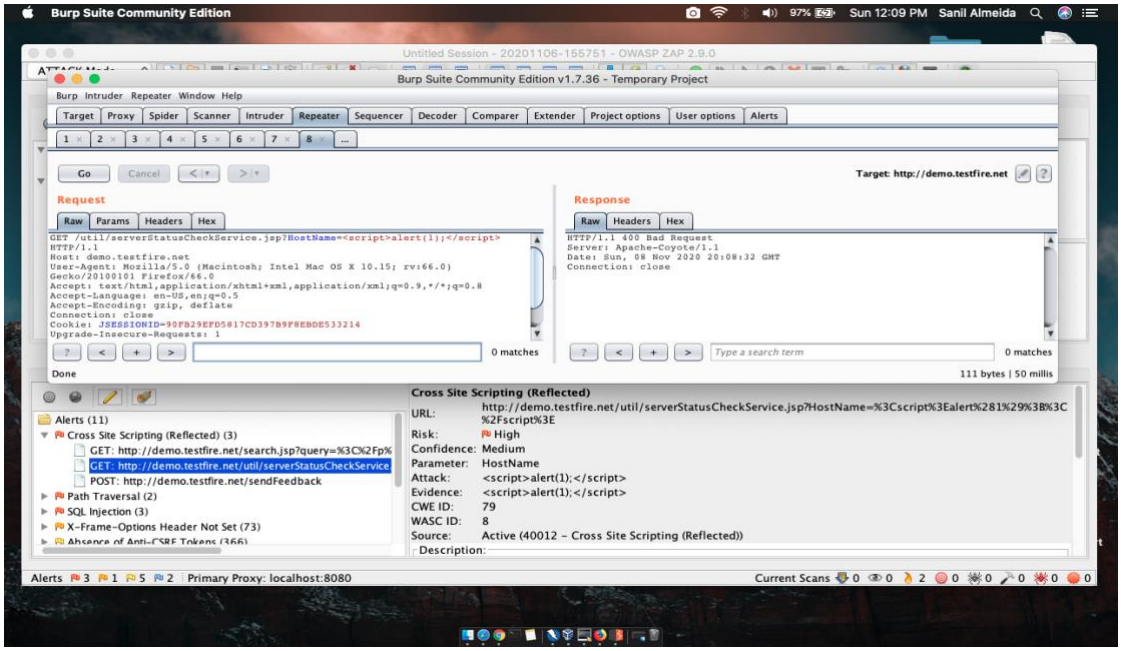
- Vulnerable URL: http://demo.testfire.net/search.jsp?query= <script>alert(1);</script>
- Risk: High
- Injection Parameter: query
- Attack: <script>alert(1);</script>
- Proof of Concept:

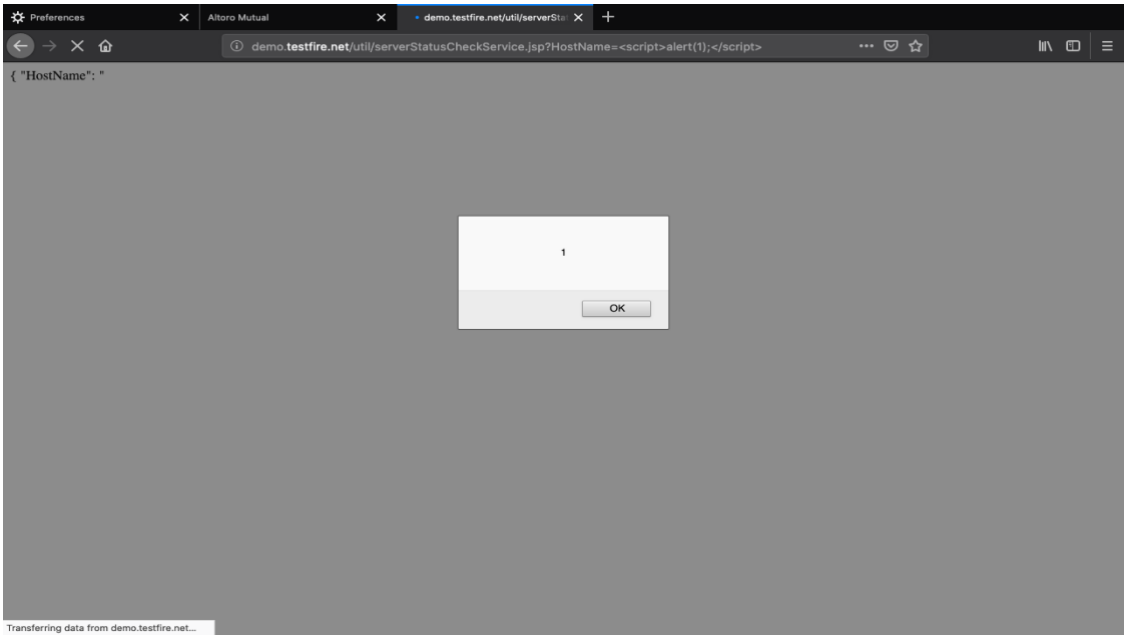




2. Attack 2

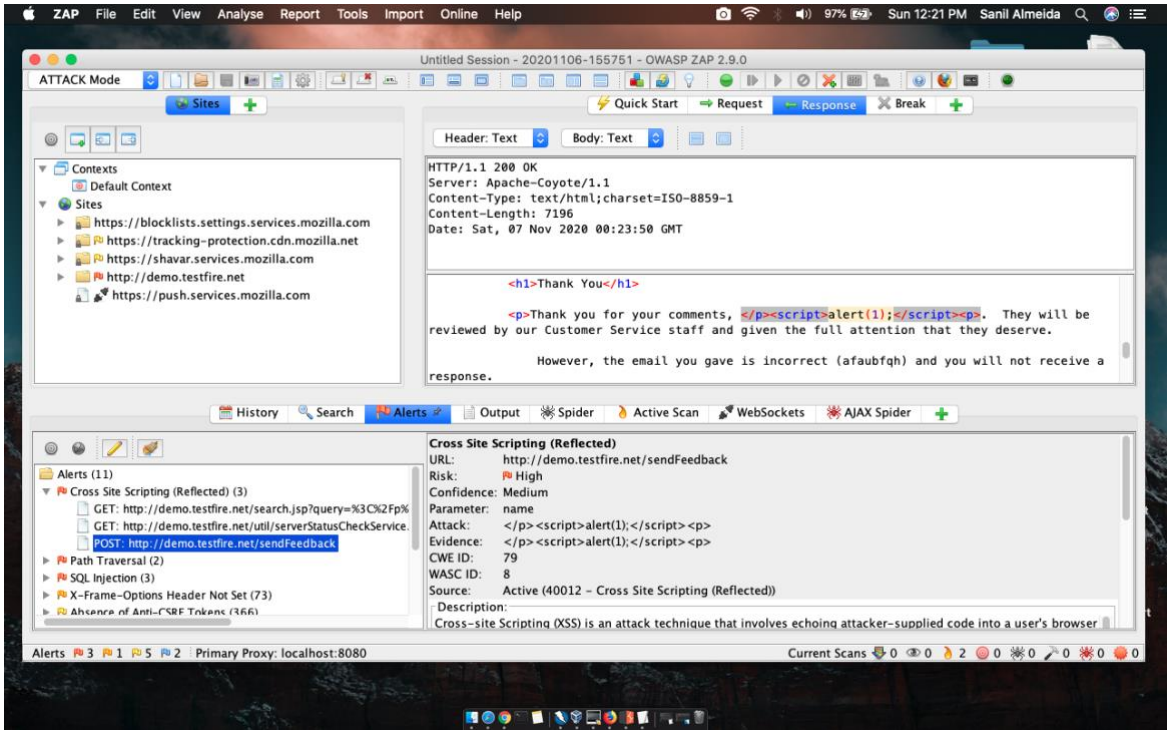
- Vulnerable URL: http://demo.testfire.net/util/serverStatusCheckService.jsp?HostName=<script>alert(1);</script>
- Risk: High
- Injection Parameter: HostName
- Attack: <script>alert(1);</script>
- Proof of Concept:

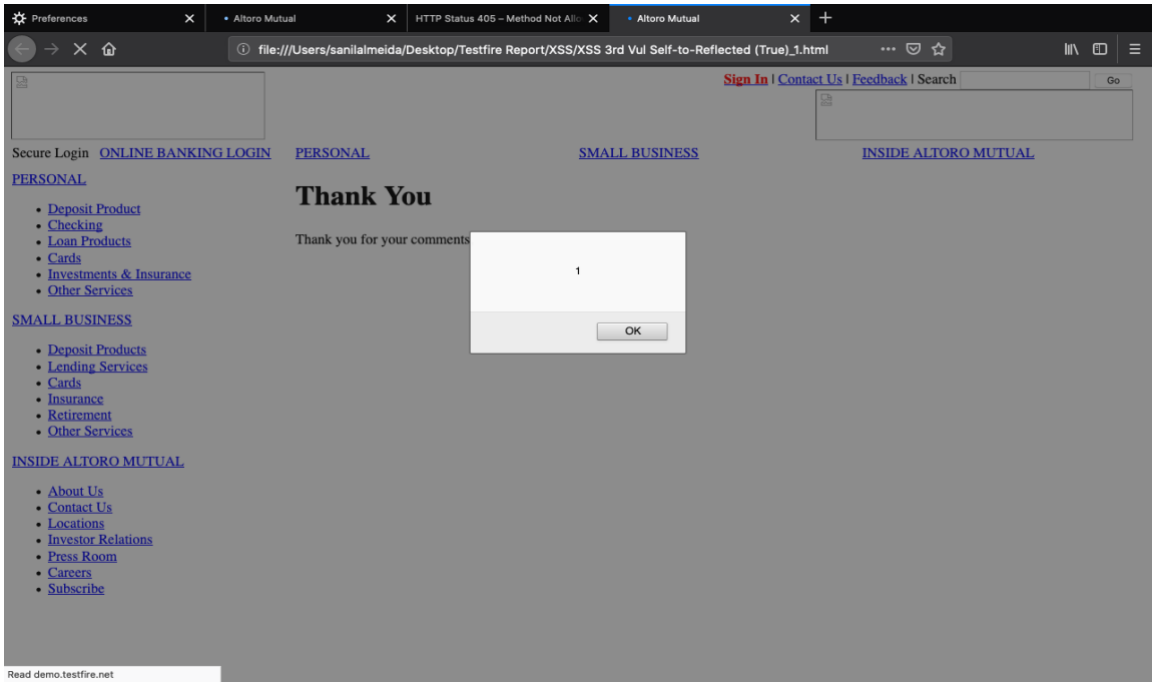




3. Attack 3

- Vulnerable URL: http://demo.testfire.net/sendFeedback
- Risk: High
- Injection Parameter: name
- Attack: <script>alert(1);</script>
- Proof of Concept: **This is a Self-Reflected XSS (The html file is attached with doc)**





4.3. Path Traversal

What is Path Traversal attack?

A path traversal attack (also known as directory traversal) aims to access files and directories that are stored outside the web root folder. By manipulating variables that reference files with “dot-dot-slash (../)” sequences and its variations or by using absolute file paths, it may be possible to access arbitrary files and directories stored on file system including application source code or configuration and critical system files.

What is the impact of Remote OS Command injection attack?

With a system vulnerable to directory traversal, an attacker can make use of this vulnerability to step out of the root directory and access other parts of the file system. This might give the attacker the ability to view restricted files, which could provide the attacker with more information required to further compromise the system.
Depending on how the website access is set up, the attacker will execute commands by impersonating himself as the user which is associated with “the website”. Therefore it all depends on what the website user has been given access to in the system.

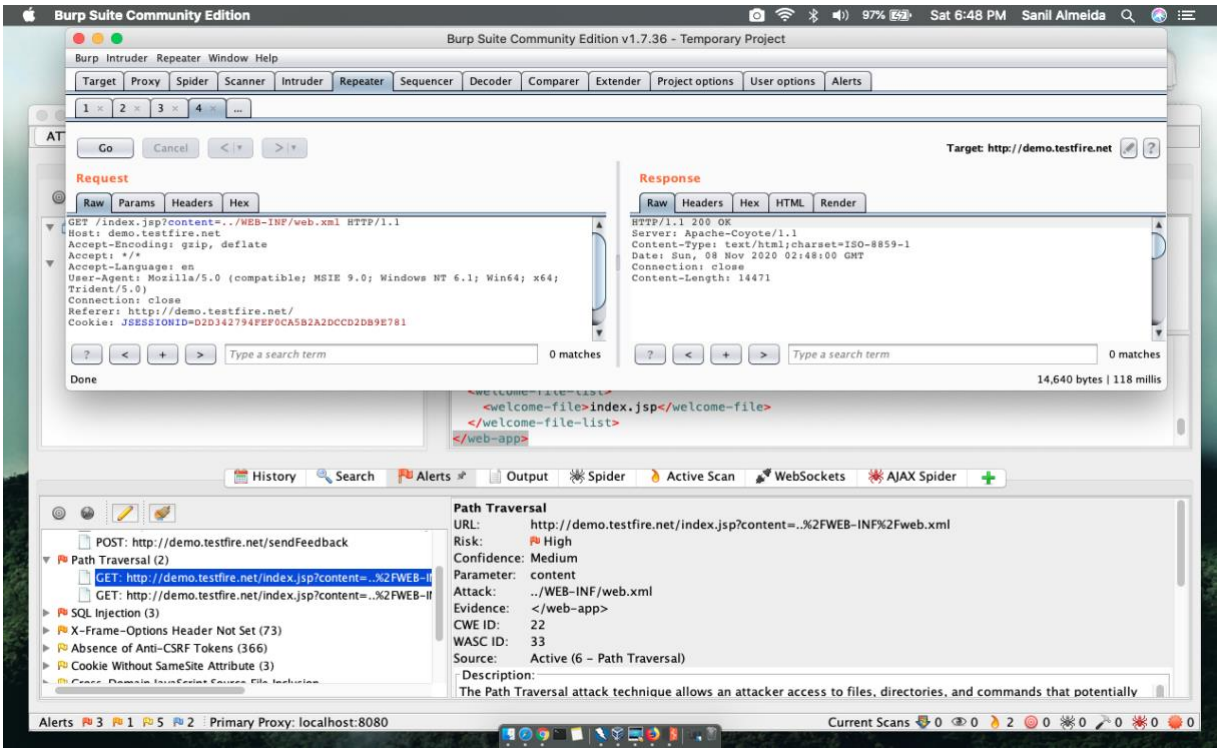
How to prevent it.

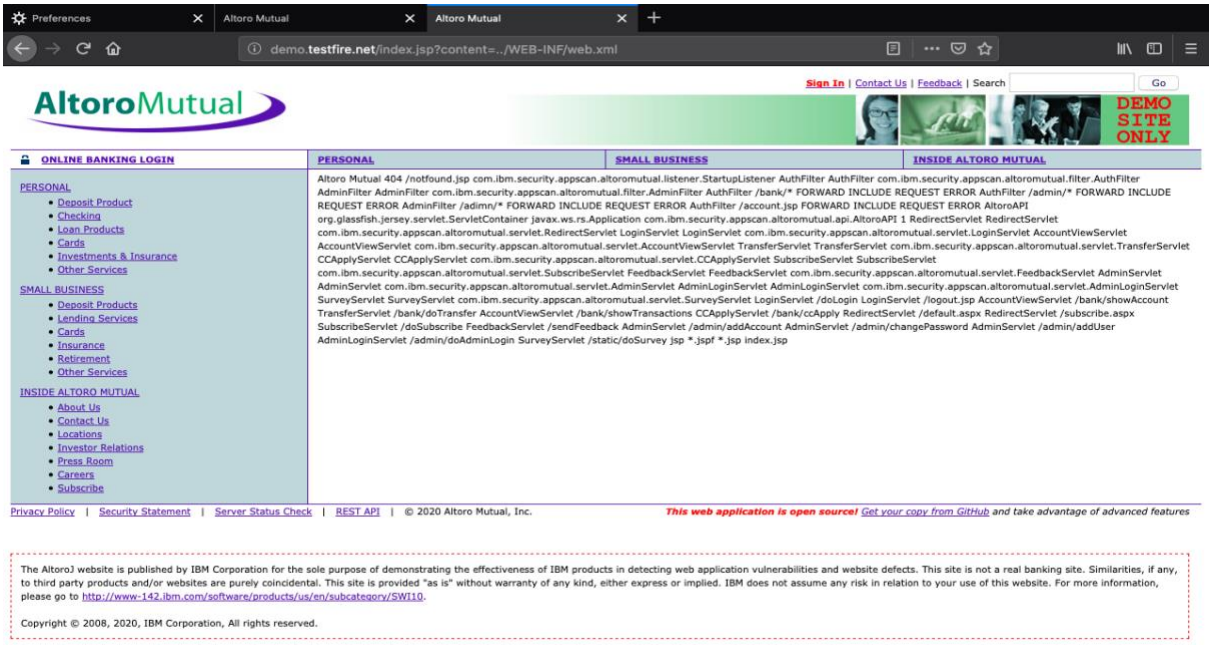
1. Prefer working without user input when using file system calls.
2. Use indexes rather than actual portions of file names when templating or using language files.
3. Ensure the user cannot supply all parts of the path – surround it with your path code.
4. Validate the user's input by only accepting known good – do not sanitize the data.
5. Use ch-rooted jails and code access policies to restrict where the files can be obtained or saved to.
6. If forced to use user input for file operations, normalize the input before using in file io API's, such as normalize().

List of vulnerable links to Path Traversal.

1. Attack 1

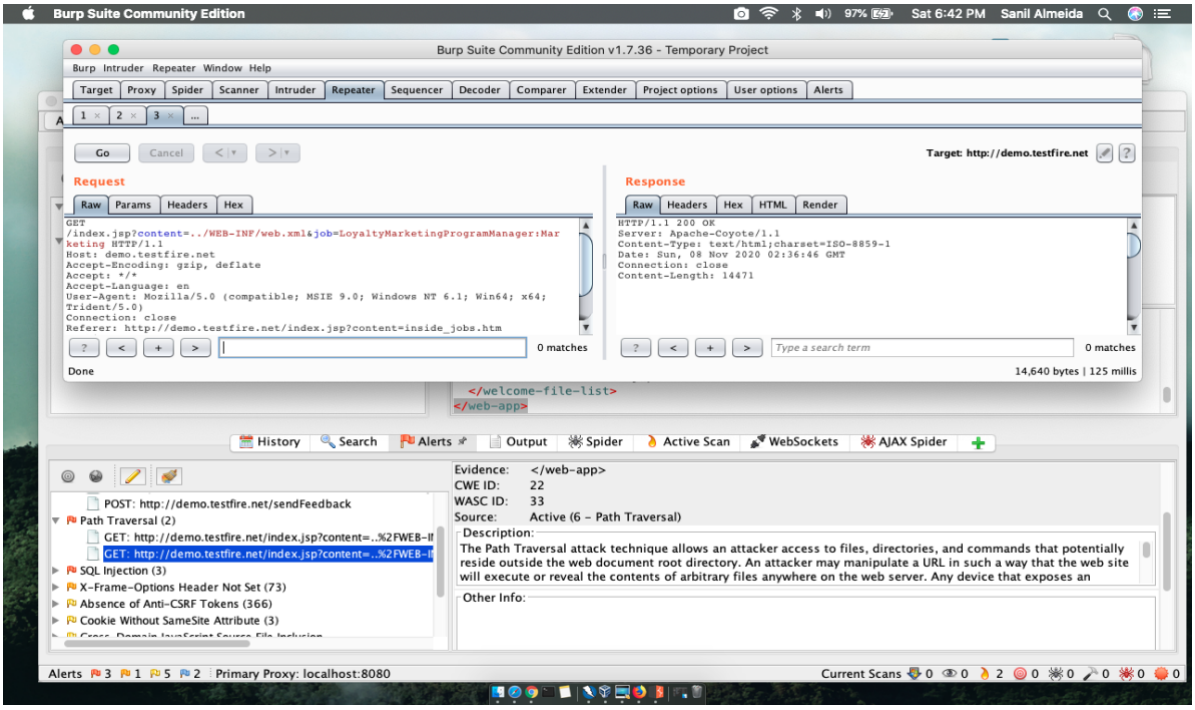
- Vulnerable URL: http://demo.testfire.net/index.jsp?content=../WB-INF/web.xml
- Risk: High
- Injection Parameter: content
- Attack: ../WEB-INF/web.xml
- Proof of Concept:

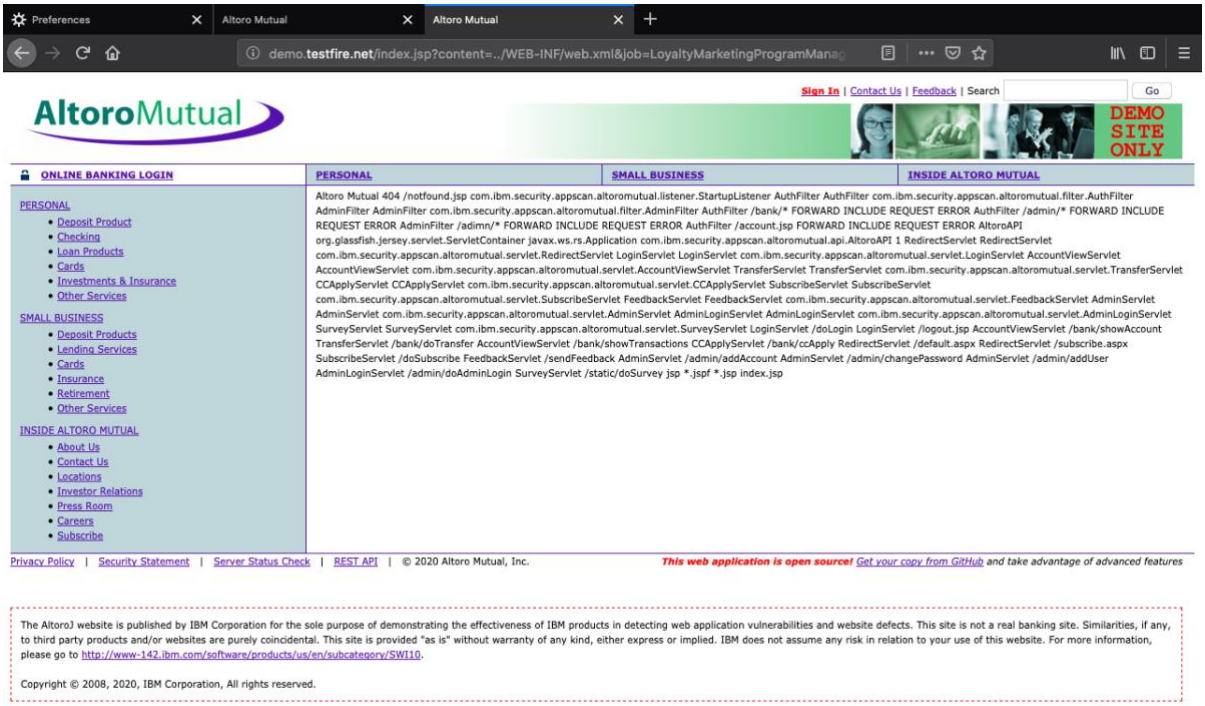




2. Attack 2

- Vulnerable URL: http://demo.testfire.net/index.jsp?content=../WEB-INF/web.xml&job=LoyaltyMarketingProgramManager:Marketing
- Risk: High
- Injection Parameter: content
- Attack: ../WEB-INF/web.xml
- Proof of Concept:





4.4. Information Disclosure

What is Information Disclosure?

Information disclosure, also known as information leakage, is when a website unintentionally reveals sensitive information to its users. Depending on the context, websites may leak all kinds of information to a potential attacker, including:

- Data about other users, such as usernames or financial information.
- Sensitive commercial or business data.
- Technical details about the website and its infrastructure.

What is the impact of Information Disclosure?

Information disclosure vulnerabilities can have both a direct and indirect impact depending on the purpose of the website and, therefore, what information an attacker is able to obtain. In some cases, the act of disclosing sensitive information alone can have a high impact on the affected parties.

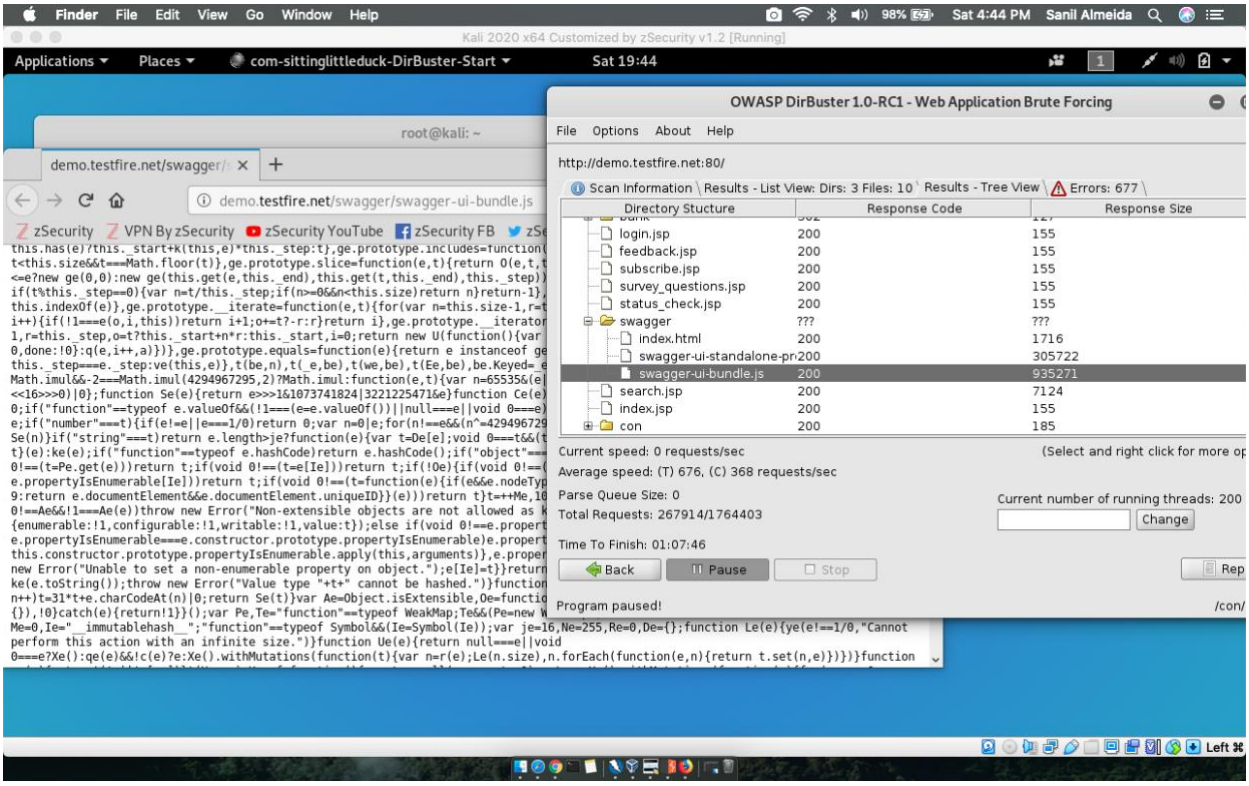
How to prevent it.

1. Make sure that everyone involved in producing the website is fully aware of what information is considered sensitive.
2. Audit any code for potential information disclosure as part of your QA or build processes. It should be relatively easy to automate some of the associated tasks, such as stripping developer comments.
3. Use generic error messages as much as possible. Don't provide attackers with clues about application behaviour unnecessarily.
4. Double-check that any debugging or diagnostic features are disabled in the production environment.
5. Make sure you fully understand the configuration settings, and security implications, of any third-party technology that you implement. Take the time to investigate and disable any features and settings that you don't actually need.

List of vulnerable links to Information Disclosure.

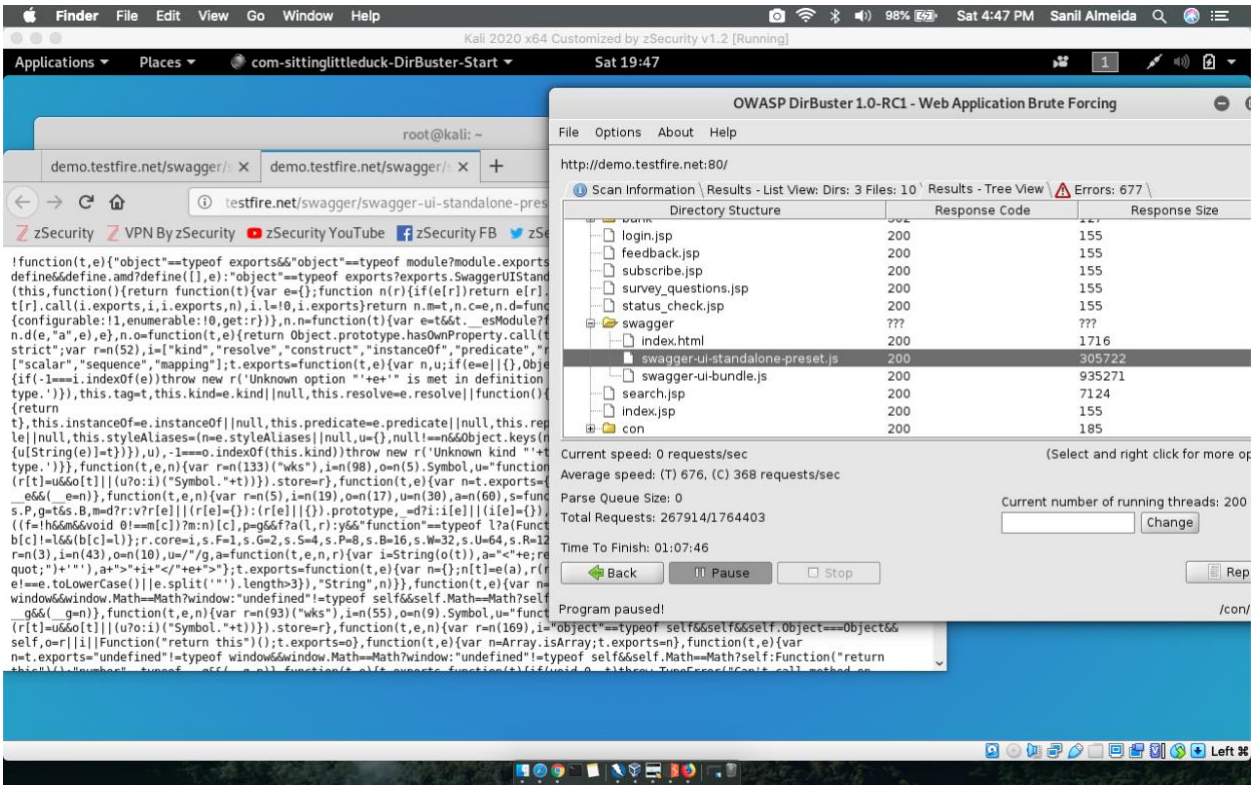
1. Attack 1

Source Code disclosure through JS file.



2. Attack 2

Source Code disclosure through JS file



4.5. References & Useful Links

- **Reports** – Reports generated by OWASP ZAP, Nikto and Netsparker were used to generate this report. (Attached with this document)
- **Reconnaissance** – Tools including but not limited to Nikto, Nmap, WafW00f and Sublist3r were used for recon.
- **Links** –
  i. https://www.netsparker.com/blog/web-security/
  ii. https://www.infocyte.com/blog/2019/02/16/
  iii. https://www.whitehatsec.com/glossary/
  iv. https://www.acunetix.com/blog/articles/
  v. https://www.esecurityplanet.com/threats
  vi. https://portswigger.net/web-security