Website

Security Testing Report

**Prepared by:**

Uday Datrak [(theudaydatrak@gmail.com)](mailto:(Uday.datrak@imaginea.com))

# **Limitations on Disclosure and Use of this Document**

The information contained within this report is considered proprietary and confidential to “Demo Technologies”. Inappropriate and unauthorized disclosure of this report or portions of it could result in significant damage or loss to Demo Technologies. This report should be distributed to individuals on a “Need-to-Know” basis only. Paper copies should be locked up when not in use. Electronic copies should be encrypted, stored offline, and and protected appropriately.

This report contains information pertaining to the Vulnerability Assessment and Penetration Test as part of the POC conducted by Security Consultant from Demo Technologies for website.

Contents

[1. Background and Scope 4](#__RefHeading___Toc17886)

[1.1 Scope 4](#__RefHeading___Toc4997)

[2. Executive Summary 5](#__RefHeading___Toc10140)

[3. Detailed Assessment Report 5](#__RefHeading___Toc5223)

[3.1 Vulnerability Description 6](#__RefHeading___Toc12867)

[3.1.1 Information transmitted in clear text 6](#__RefHeading___Toc19879)

[3.1.2 Stored Cross Site Scripting 7](#__RefHeading___Toc30589)

[3.1.3 Weak Password Policy 9](#__RefHeading___Toc13606)

[3.1.4 No Anti Automation 10](#__RefHeading___Toc32030)

[3.1.5 Auto complete is enabled 11](#__RefHeading___Toc19215)

[3.1.6 HTTP Response reveals version information 12](#__RefHeading___Toc13729)

# Background and Scope

As part of the proof-of-concept for security testing, Security testing has performed a vulnerability assessment for website URL ‘[http://localhost](http://localhost/)’.

## 1.1 Scope

* The scope of this test was limited to the following:
* Perform a Vulnerability Assessment on the following Web Application URLs:
  + [http://localhost](http://localhost/)
* The vulnerability assessment performed is an indicative assessment rather than an exhaustive assessment. With sufficient time and effort more number of vulnerabilities can be unearthed.
* The Security Assessment Technical report contains the vulnerabilities identified and categorized based on severity, recommendations to mitigate the vulnerabilities.

Below are list of security issues will be tested against the websites manually.

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No** | **Security Test Cases** | **S.No** | **Security Test Cases** |
| 1 | SQL Injection | 42 | Session identifier can be preset |
| 2 | Vulnerable versions | 43 | Information Disclosure through error messages |
| 3 | Information transmitted in clear text | 44 | Password cached in browser memory |
| 4 | Older SSL version supported | 45 | Weak Session Expiry |
| 5 | Server supports weak SSL ciphers | 46 | Application code has references to third party websites |
| 6 | Default Login Accounts are enabled on the application | 47 | Session Identifier is stored in Persistent cookie |
| 7 | Authentication Bypass | 48 | Session ID value not changed on authentication |
| 8 | Authorization bypass | 49 | Session Hijacking |
| 9 | Weak Authorization Control | 50 | Auto complete is enabled |
| 10 | Stored Cross Site Scripting (XSS) | 51 | Debug method enabled |
| 11 | HTTP Parameter pollution attack | 52 | No Change Password page |
| 12 | SSI Injection | 53 | Vulnerable Change Password feature |
| 13 | Xquery Injection | 54 | Password field type is set to text |
| 14 | Expression Language Injection | 55 | Cookie based Session management is not implemented |
| 15 | XXE Injection | 56 | Sensitive data present in browser cache |
| 16 | Cross Site Request Forgery (CSRF) | 57 | Client-Side validation bypass |
| 17 | Directory Traversal vulnerability | 58 | CrossDomain.xml File Has Allow-all Policy |
| 18 | Vulnerable File Upload | 59 | Unnecessary HTTP methods in use |
| 19 | Weak Password Policy | 60 | Sensitive information passed in GET request |
| 20 | Forceful Browsing | 61 | ‘Remember Me’ feature of the application can reveal sensitive information |
| 21 | Improper implementation of Functionality | 62 | Application Log files stored in local machine |
| 22 | Application returns password in response in plain text | 63 | SSL certificate error |
| 23 | Vulnerable Forgot Password | 64 | Backup/Default/test files are accessible remotely |
| 24 | Insecure Password storage | 65 | HTTP Response reveals version information |
| 25 | Application is vulnerable to HTTP Response Splitting | 66 | Session cookie attributes are not set securely |
| 26 | Application is vulnerable to URL Redirection | 67 | Cross Frame Scripting |
| 27 | Basic Authentication is used | 68 | Weak Session Identifier |
| 28 | Server/OS files are accessible remotely | 69 | User Enumeration |
| 29 | Source Code accessible remotely | 70 | Logout button is not displayed on all pages |
| 30 | Password displayed in plain text | 71 | Concurrent Logins |
| 31 | Reflected Cross Site Scripting (XSS) | 72 | Directory listing is enabled |
| 32 | Session is not invalidated after Logout | 73 | Possibly vulnerable to SSLv3/TLS Renegotiation Stream Injection |
| 33 | Session Puzzling Vulnerability | 74 | Possibly vulnerable to TLS/SSL Beast attack |
| 34 | FrontPage extensions are used | 75 | Possibly vulnerable to TLS/SSL Crime Information Leakage attack |
| 35 | No Access control restriction to file path | 76 | Information Disclosure through comments in the page source |
| 36 | Internal Pages of the application are not restricted | 77 | Web Server robots.txt Information Disclosure |
| 37 | Password and username stored on client machine | 78 | Password is not encrypted |
| 38 | Admin interface accessible over the internet | 79 | Weak CAPTCHA |
| 39 | No Anti Automation | 80 | Weak Cryptographic Hash is used |
| 40 | Possibly Cross Site Scripting (XSS) vulnerability in Flash files | 81 | Content Security Policy is set to default |
| 41 | Weak Input Validation | 82 | HTTP Strict Transport Security (HSTS) header is not set |
|  |  | 83 | IOS Security Test cases related to Thumb and Device |

# Executive Summary

Our organization performed a Vulnerability Assessment on website URL ‘[http://localhost](http://localhost/)’. Security testing performed the tests from 11/07/2017 to 06/08/2017 at Demo Technologies, Hyderabad premises. The tests were carried out in conformance with the scope, and agreed upon testing schedule. The tests were completed without any incidents in terms of disruption to IT infrastructure or loss of information.

In total - 3 critical, 4 high, 19 medium and 4 low vulnerabilities were found during the security test. More information on the details of these vulnerabilities can be found in section III.

# Detailed Assessment Report

The objective of this project was to identify possible attacks that a malicious attacker might launch, in order to gain access to a web application and/or other devices, and to provide recommendations on how the risks associated with these can be mitigated. The assignment focused on external threats and assessed the effectiveness of the present security posture.

The target to be analyzed was as follows:

* + [http://localhost](http://localhost/)

# 3.1 Vulnerability Description

### XSS

#### Detailed Description:

The Test Team observed that HTTP plain text protocol is implemented to transmit application data. The application is accessible over both HTTP and HTTPS. After authentication if the user visits the HTTP website then application doesn't re-enforce HTTPS.

**Impact:**

This configuration error can aid in sniffing and disclosure of sensitive data such as user’s login credentials, or personal data, resulting in information disclosure.

#### Screenshots:

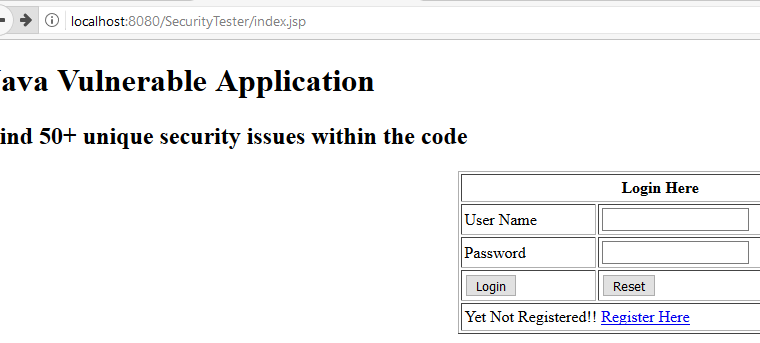


Figure 1: Application accessible over HTTP

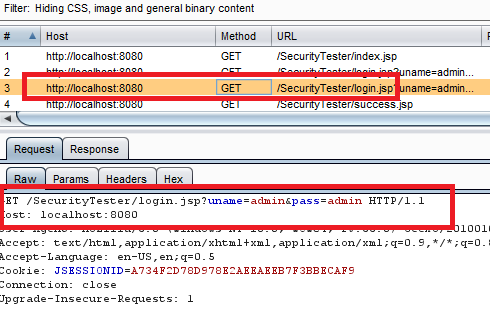


Figure 2: HTTP requests.

**Recommendations:**

It is recommended to implement SSL for transmitting all sensitive or personal information. The application should be accessible only over HTTPS. Any HTTP request for a page should be redirected to HTTPS.

Additionally, after implementing SSL, ensure that the web server does not support SSL v2.0 or older SSL versions. Only SSL 3.0 and TLS 1.0 should be supported.

Also, disable weak SSL ciphers which are less than 128 bits. Only SSL ciphers which are at least 128 bits in strength should be supported.

**References:**

<https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet>

### Stored Cross Site Scripting

#### Detailed Description:

The test team observed that the web application is vulnerable to persistent (stored) Cross Site Scripting (XSS) vulnerability in the application. This vulnerability may allow an attacker or an intruder to issue malicious java-scripts, place hidden frames or deceptive links to unrelated or business competition sites and steal authentication cookies.

Vulnerable Parameters identified are First name, Last name.

In the current scenario logged-in Attacker can get the session IDs of logged-in users without asking them to click on links as the script directly executes on home page of logged-in users.

**Impact:**

This may result in Session Hijacking, disclosure of sensitive data, log-in credentials, and/or more serious data manipulation.

#### Screenshots:

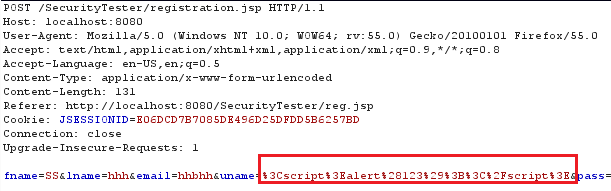


Figure 1: Request to store the JavaScript on the server.

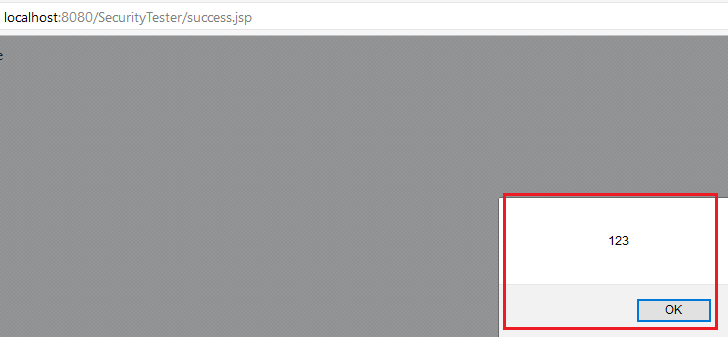


Figure 2: Script execution response for above request

**Recommendations:**

It is recommended that secure coding and design practices be implemented.

Validation of all headers, cookies, query strings, form fields, and hidden fields (i.e., all parameters) against a rigorous specification of what should be allowed. Special characters such as ‘ “ < > . , \* # % & / \ |( ) : ; - etc. should be filtered out wherever not expected.

In case special characters are required, these should be encoded using an appropriate encoding mechanism (e.g., HTML encoding) to ensure that these are not interpreted by the browser.

**Note:** Recommendation should be implemented throughout the application.

**References:**

<https://www.owasp.org/index.php/XSS_(Cross_Site_Scripting)_Prevention_Cheat_Sheet>

### Weak Password Policy

#### Detailed Description:

The Test Team observed that the application is implemented with weak password policy. The application accepts the following weak passwords:

• Password of length less than 8 characters are allowed

• Passwords without digits or Special characters are allowed

This finding is applicable for the following pages in the application:

• Change Password Page

Also, the application does not ask the new users to change their passwords when they log in to the application for the first time. Since the passwords are created by admin, it is known to both the administrator and the user.

**Impact:**

Weak password policy makes the user of the application vulnerable to brute force attack, which leads to compromise of user accounts.

#### Screenshots:

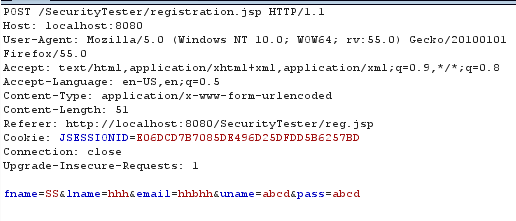


Figure 1: Weak Password Policy

**Recommendations:**

It is recommended to implement a strong password policy as follows:

Password Length:

• Passwords for user accounts must be at least 6 characters long.

• Passwords of privileged accounts must be at least 8 characters long.

Password Complexity:

1. Passwords must be selected to combine at least 2 kinds of characters such as alphabet and number as a general rule. (Using only numbers or alphabet is prohibited. In case the system cannot allow a combination of alphabet and numbers, the following types of passwords are prohibited.

• All-numeric or all-alphabetical groups (e.g., kkkkkkkk?5555555).

• Consecutive identical numeric and alphabetical characters (e.g., 9876543?HIJKLMNO).

2. Privileged account passwords must be a combination of uppercase alphabet characters, lower case alphabet characters, numeric characters and special characters.

3. Passwords must not be easily guessed. The following types of passwords are prohibited:

• Your ID or a sequence that includes characters of your ID (e.g., not to use suzukia, if your ID is suzuki).

• When an e-mail address is registered, the sentence before @ (e.g., not to use xsuzuki, if the e-mail address is xsuzuki@xxx.xxxx.xxx).

• Last name, first name, initial of name, or mixture of them.

Password History:

Previously used passwords must not be re-used.

In particular, the following are prohibited when changing passwords.

• Substituting only a single character in the old password (e.g. ac8g?qo1 ? ac8g?qo2).

• Using the sequence of the old password backwards (e.g. ac8g?qo1 ? loq?g8ca).

The Password Policy should be implemented on the following pages:

• Change Password Page

• Password Reset Page (available to admin users)

• Registration Page

• User Creation page

Also, the application should ask the new users to change their passwords when they log in to the application for the first time.

Note: The password policy should be validated at the server side.

**References:**

<http://blogs.msdn.com/b/kaushal/archive/2012/01/21/fixing-the-beast.aspx>

### No Anti Automation

#### Detailed Description:

The Test Team observed that no anti-automation mechanisms like CAPTCHA or any other challenge-response based security measures is available in the application to differentiate between humans and computers.

The following pages in the application do not have any anti-automation mechanism like CAPTCHA:

• Login Page

• Registration Page

• Forgot Password page

**Impact:**

This vulnerability can be exploited by an attacker to replay multiple automated http requests causing an increase in the consumption and utilization of bandwidth, memory and processor of the target host. This could result in a possible denial of service (DOS) attack.

The attacker can also perform brute force attacks on the Login page.

#### Screenshots:



Figure 1: No CAPTCHA on the page

**Recommendations:**

The publicly available forms should be enabled with anti-automation mechanisms like CAPTCHA which is a challenge-response test that can differentiate between humans and computers. Implementation of CAPTCHA or Turing mechanisms will help mitigate the risk of automated attacks.

Note: CAPTCHA should be implemented in the following pages in the application:

• Login Page

• Registration Page

• Forgot Password page

**References:**

[http://projects.webappsec.org/w/page/13246938/Insufficient%20Anti-automation](http://projects.webappsec.org/w/page/13246938/Insufficient Anti-automation)

### Auto complete is enabled

#### Detailed Description:

The Test Team observed that 'Autocomplete' is enabled or set to 'ON' for 'Username' fields in the application.

Note: Autocomplete attribute is also enabled for different form fields located on many pages in the application. The following list is example for a few pages where Autocomplete is enabled. Example snapshots showing the vulnerability have been provided in the ‘Screenshots’ section for only one of the vulnerable page. Recommendation should be implemented on all the pages.

• Forgot password page: form fields with personal information such as email address.

**Impact:**

In the event that the ‘Autocomplete’ feature is set to ‘ON’, any user accessing the application from the computer or kiosk system on which the application was accessed previously will be prompted with the username and other personal information previously used. This can allow an attacker to extract valid usernames, passwords and other personal information from public computers or multi-user environments.

#### Screenshots:

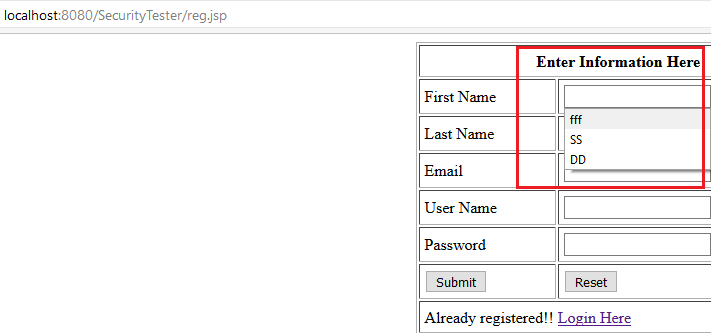


Figure 1: Autocomplete enabled

**Recommendations:**

It is recommended that 'Autocomplete' attribute be turned OFF for such sensitive and personal data fields, to minimize the chances of revealing valid usernames and personal information and resulting in unauthorized access.

**References:**

<https://www.netsparker.com/web-vulnerability-scanner/vulnerability-security-checks-index/auto-complete-enabled/>

### HTTP Response reveals version information

#### Detailed Description:

It was observed that HTTP response reveals banner Information of the server in use. The following version information is revealed in HTTP responses:

• Apache/8.0.36

**Impact:**

In the current scenario, an attacker may be able to find known vulnerabilities in the installed version of the service to plan an attack and exploit those vulnerabilities further.

#### Screenshots:



Figure 1: Version revealed in HTTP response

**Recommendations:**

It is recommended that server side hardening be implemented so that no information like service running, version detail, etc. is revealed to the attacker through HTTP response or accessed by banner grabbing tools.

**References:**

<http://www.tecmint.com/linux-server-hardening-security-tips/>

1. **Severity Classification**

Below is a summary of the factors which illustrate types of vulnerabilities usually resulting in a specific severity level.

**Severity Level - Critical:**

Vulnerabilities that score in the critical range usually have most of the following characteristics:

1. Exploitation of the vulnerability results in root-level compromise of servers or infrastructure devices.
2. The information required in order to exploit the vulnerability, such as example code, is widely available to attackers.
3. Exploitation is usually straightforward, in the sense that the attacker does not need any special authentication credentials or knowledge about individual victims, and does not need to persuade a target user, for example via social engineering, into performing any special functions.
4. For critical vulnerabilities, is advised that you patch or upgrade as soon as possible, unless you have other mitigating measures in place. For example, if your installation is not accessible from the Internet, this may be a mitigating factor.

**Severity Level - High:**

Vulnerabilities that score in the high range usually have some of the following characteristics:

1. The vulnerability is difficult to exploit.
2. Exploitation does not result in elevated privileges.
3. Exploitation does not result in a significant data loss.

**Severity Level - Medium:**

Vulnerabilities that score in the medium range usually have some of the following characteristics:

1. Denial of service vulnerabilities that are difficult to set up.
2. Exploits that require an attacker to reside on the same local network as the victim.
3. Vulnerabilities that affect only nonstandard configurations or obscure applications.
4. Vulnerabilities that require the attacker to manipulate individual victims via social engineering tactics.
5. Vulnerabilities where exploitation provides only very limited access.

**Severity Level - Low:**

Vulnerabilities in the low range typically have very little impact on an organization's business. Exploitation of such vulnerabilities usually requires local or physical system access.