Chinmaya Mission Washington Regional Center Application

Vulnerability Assessment and Penetration Test

June 2020

# 

**Naga Thota**

**Security Consultant**

Contents

[1](#_Toc34123257)

[Executive Summary 3](#_Toc34123258)

[Methodology 4](#_Toc34123259)

[Scope 4](#_Toc34123260)

[User Roles: 4](#_Toc34123261)

# **Executive Summary**

Srini K engaged Naga Thota to perform a penetration test of Chinmaya Mission Washington Regional Center application. The purpose of this assessment was to assess the overall security posture of the application from a white-box perspective. This includes determining the application’s ability to resist common attack patterns and identifying vulnerable areas in the internal or external interfaces that may be exploited by a malicious user.

I conducted my testing in accordance with the OWASP Top 10.

I found 1 high, 2 medium and 2 low risk vulnerabilities as shown below.

|  |  |
| --- | --- |
| **Risk** | **Issues** |
| High | OTP Bypass |
| Medium | Directory Listing |
| Medium | Frameable response (potential Clickjacking) |
| Low | Information Disclosure |
| Low | Application Exception |

## Methodology

Naga Thota was engaged to perform a time-boxed manual security assessment against the Techni Trace application. This assessment involved a deep automated scan using automated scanning tools to discover common vulnerabilities, as well as manual testing. Manual testing includes validation of all issue types covered under the automated scan as well as checks for problems not typically found by automated scanners such as authentication, authorization and business logic flaws.

##### Scope:

* <https://regcore.cmwrcregistration.org/site/login>
* <https://regqa.cmwrcregistration.org/site/login>

**Definitions**

|  |  |
| --- | --- |
| High | Weaknesses classified as High can be exploited with little effort by an attacker. They have a major negative impact on the tested system, its users and data, or the system environment. |
| Medium | Weaknesses classified as Medium can be exploited with medium effort by an attacker. They have a medium negative impact on the tested system, its users and data, or the system environment. |
| Low | Weaknesses classified as Low can be exploited with great effort by an attacker. They have little negative impact on the tested system, its users and data, or the system environment. |
| Information | Observations classified as Information are usually no weaknesses. Examples of these observations are unusual configurations and possibly unwanted behavior of the tested system. |

## OTP Bypass

User can bypass the OTP verification needed while login the application. User can give a random number and intercept the OTP request If wrong OTP is provided then the response shows the numeric value "1" is the OTP in this case. If we changed the numeric value "1" to "0" the OTP validated and bypassed.

Issues presents in **Sign in** and **Sign up** forms.

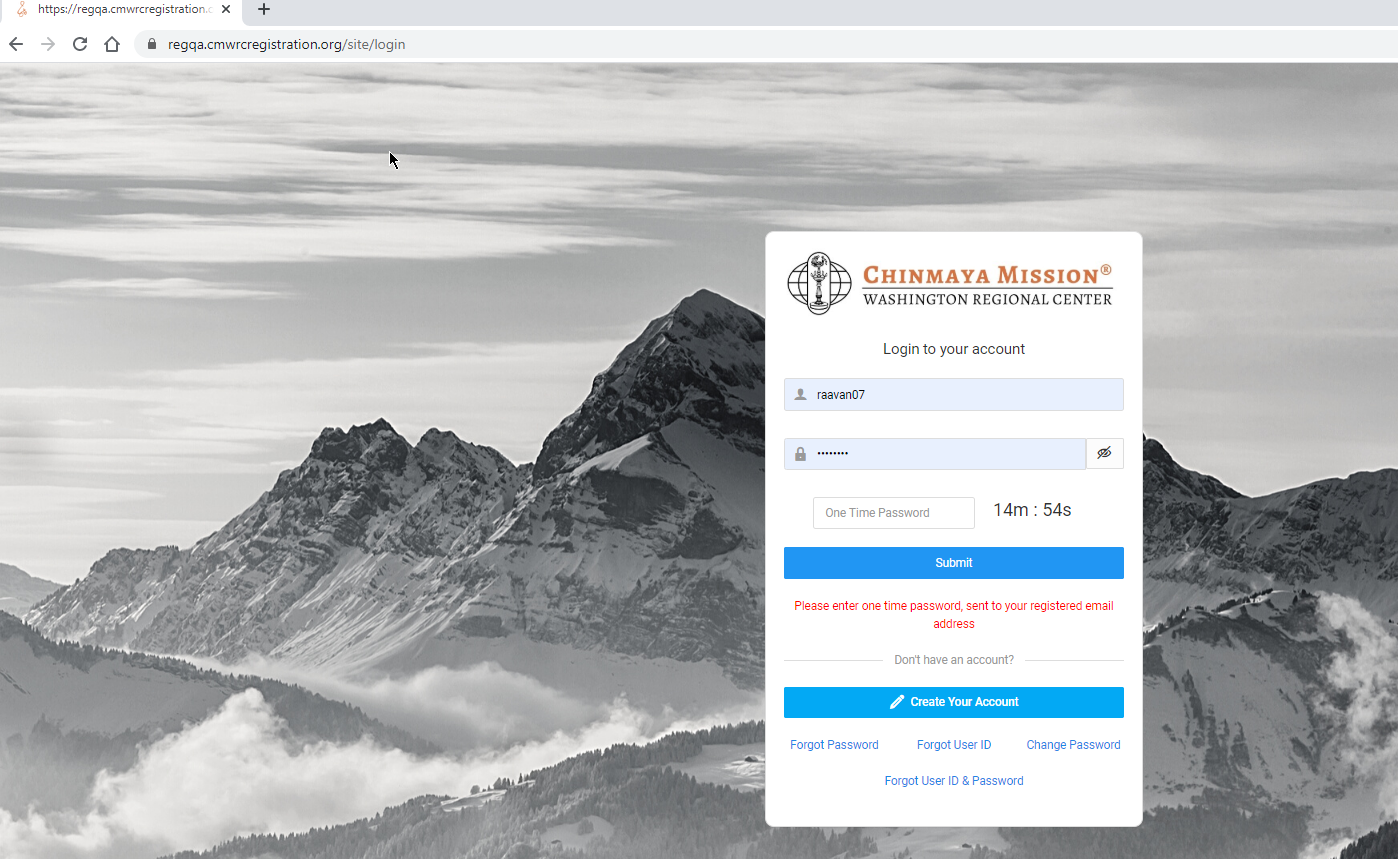
**Path**:

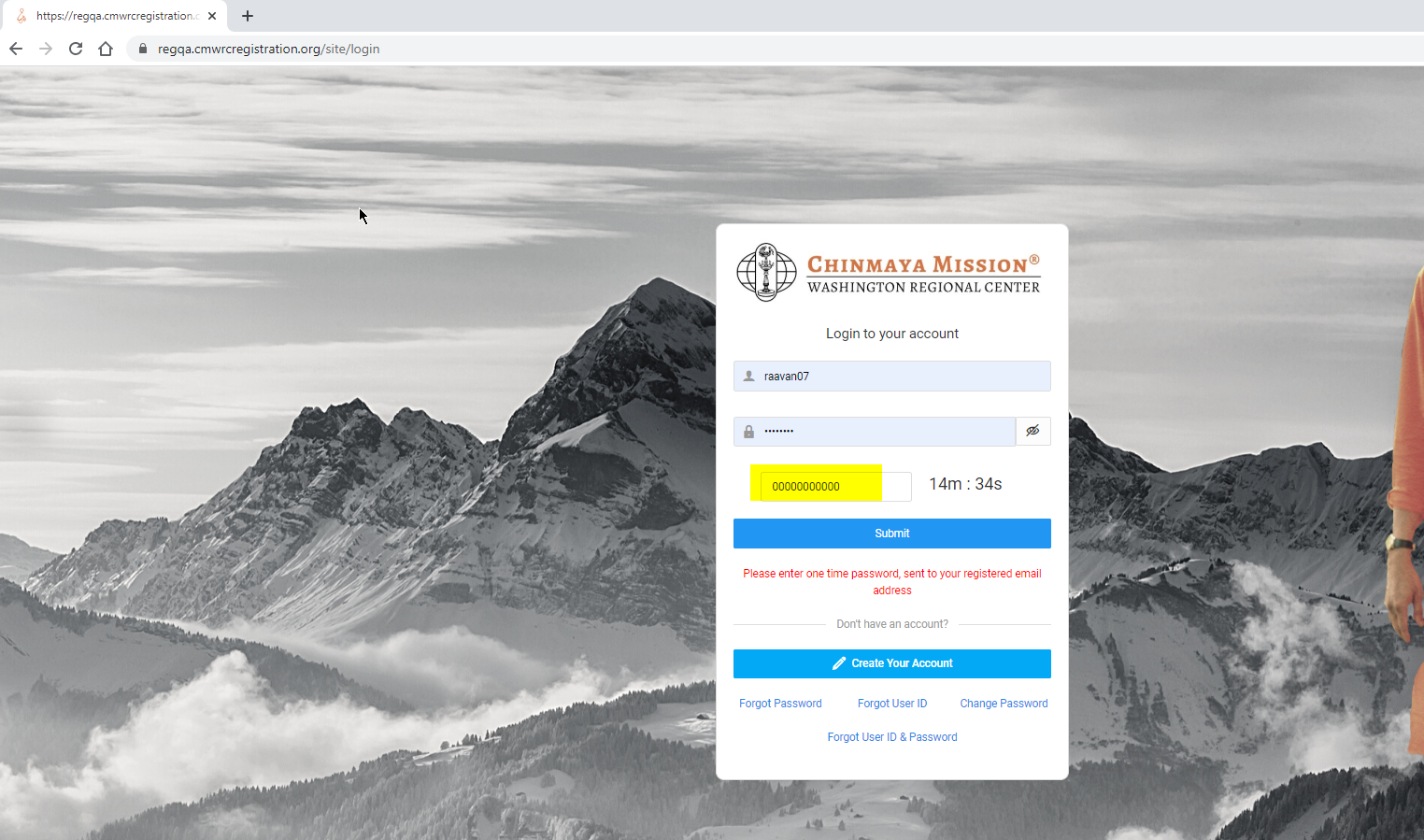
* <https://regqa.cmwrcregistration.org/site/login>
* <https://regcore.cmwrcregistration.org/site/login>

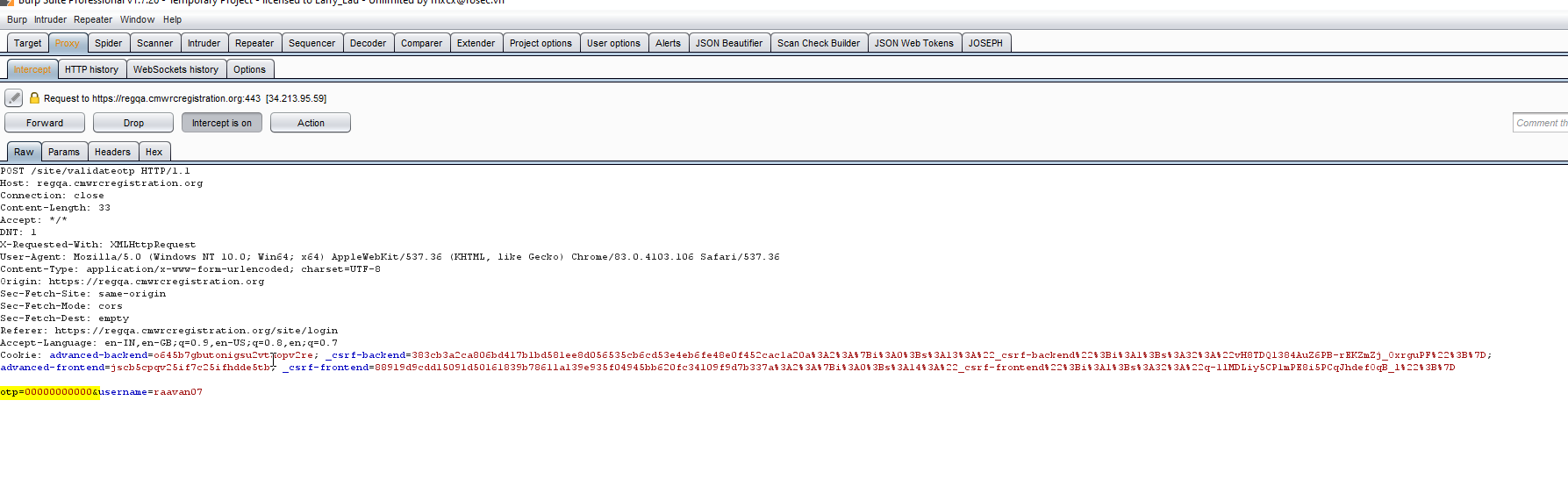
**Steps to reproduce**

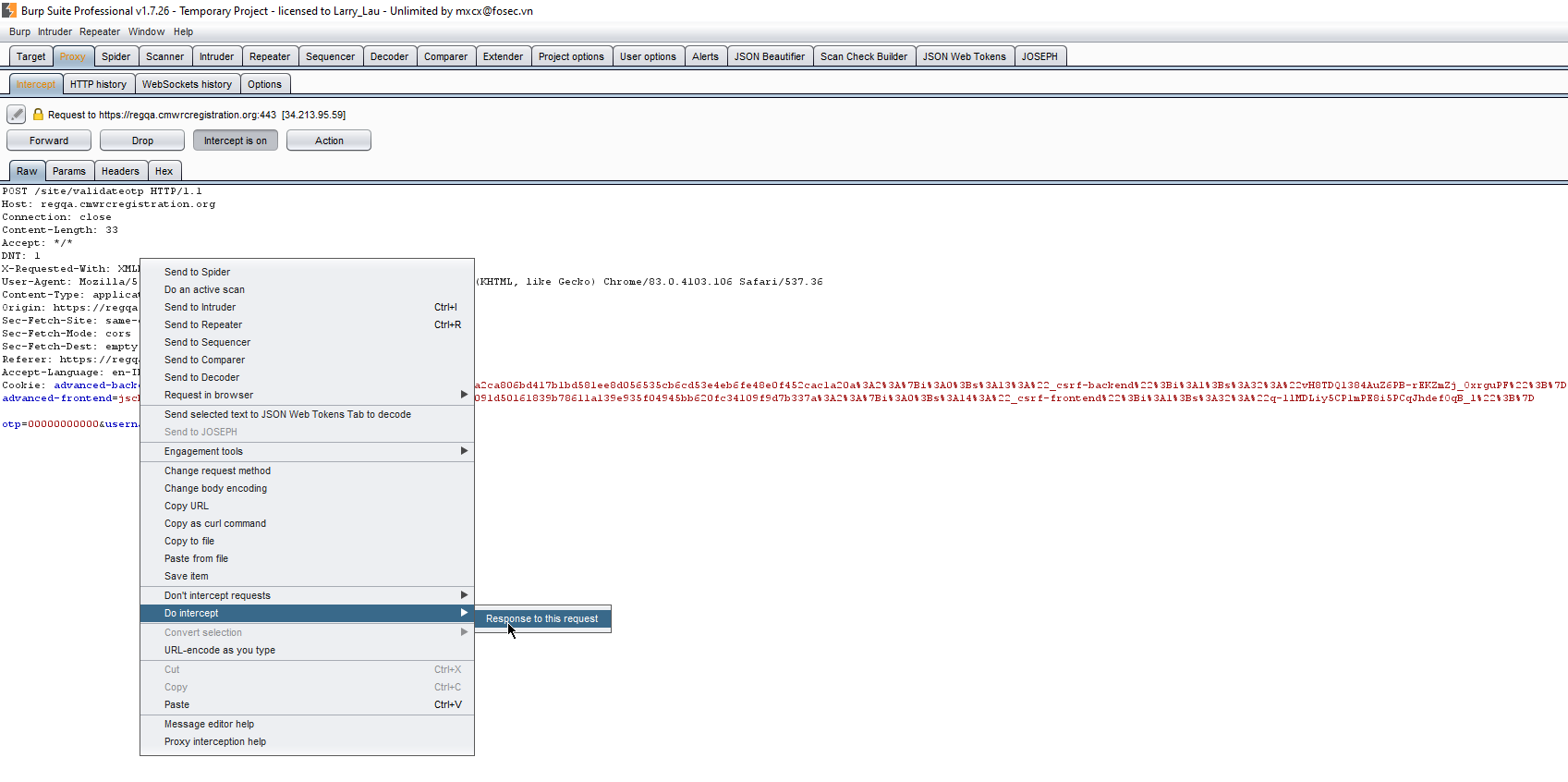
* Log in to the applications - <https://regqa.cmwrcregistration.org/site/login>
* Capture the server response using Burp suite proxy tools
* Send the any invalid OTP
* Intercept the server response and change the value “1” to “0”

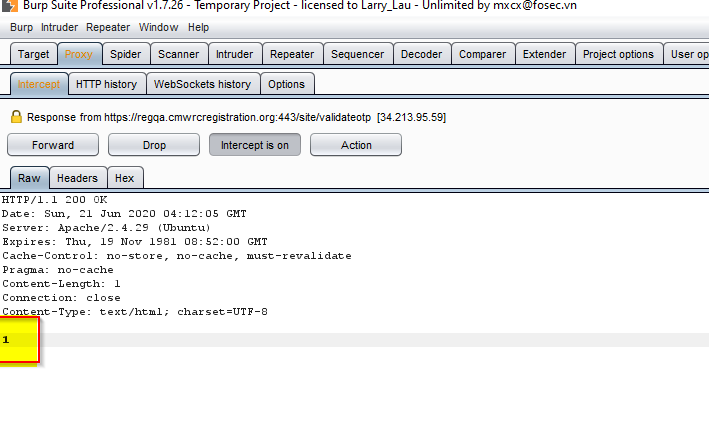
**Screenshots**

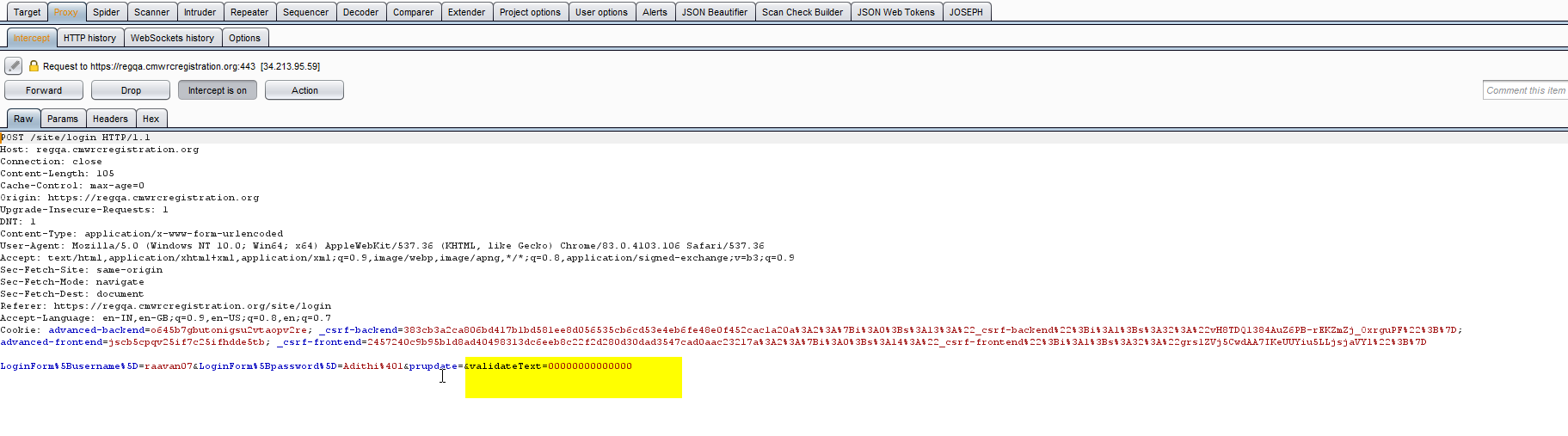


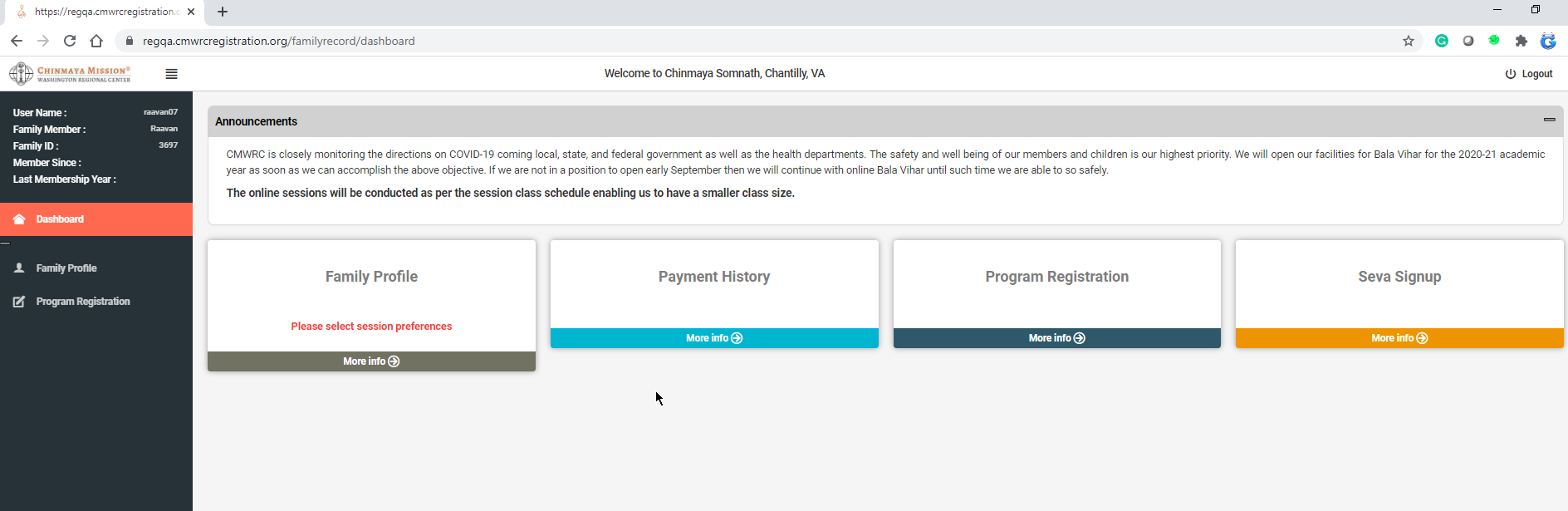












## Directory Listing

**Description**

Web servers can be configured to automatically list the contents of directories that do not have an index page present. This can aid an attacker by enabling them to quickly identify the resources at a given path, and proceed directly to analyzing and attacking those resources. It particularly increases the exposure of sensitive files within the directory that are not intended to be accessible to users, such as temporary files and crash dumps.

Directory listings themselves do not necessarily constitute a security vulnerability. Any sensitive resources within the web root should in any case be properly access-controlled, and should not be accessible by an unauthorized party who happens to know or guess the URL. Even when directory listings are disabled, an attacker may guess the location of sensitive files using automated tools.

**Remediation**

There is not usually any good reason to provide directory listings, and disabling them may place additional hurdles in the path of an attacker. This can normally be achieved in two ways:

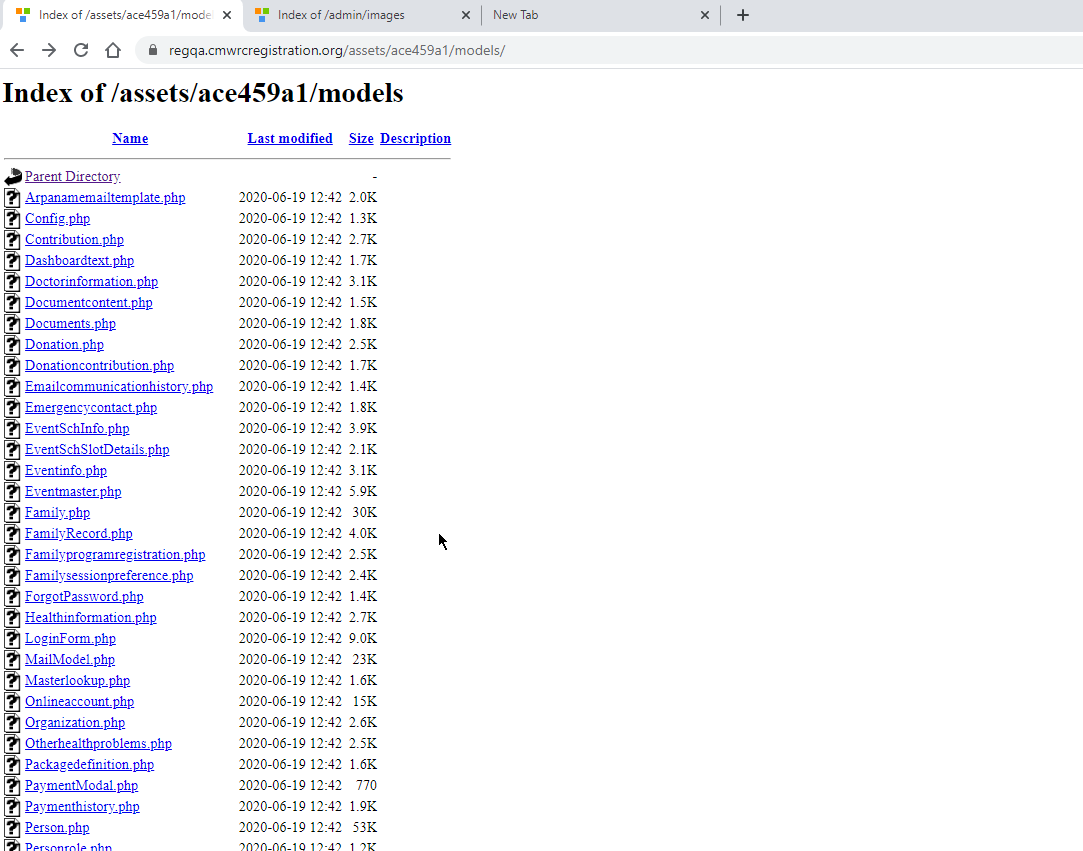
Configure your web server to prevent directory listings for all paths beneath the web root;

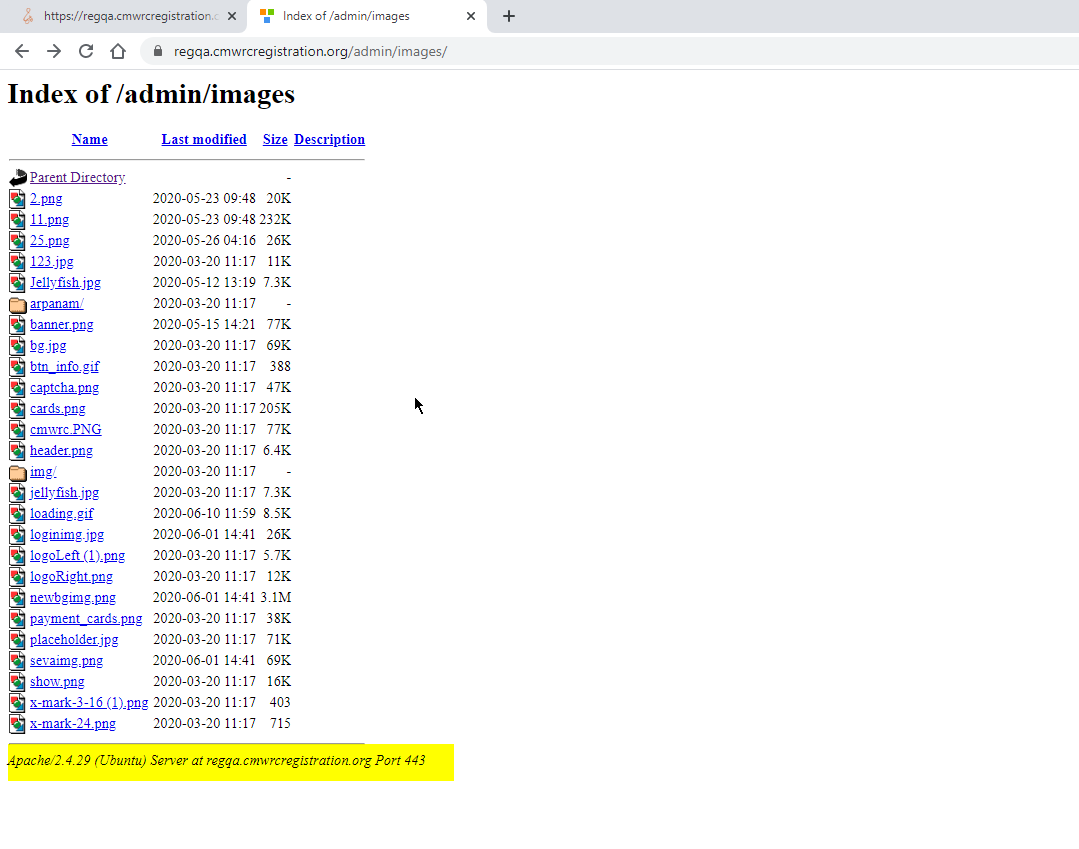
Place into each directory a default file (such as index.htm) that the web server will display instead of returning a directory listing.

**Path**

* <https://regqa.cmwrcregistration.org/assets/ace459a1/models/>
* <https://regqa.cmwrcregistration.org/admin/images/>
* <https://regcore.cmwrcregistration.org/admin/images/>
* <https://regcore.cmwrcregistration.org/assets/c2268488/config/>

**Screenshots**





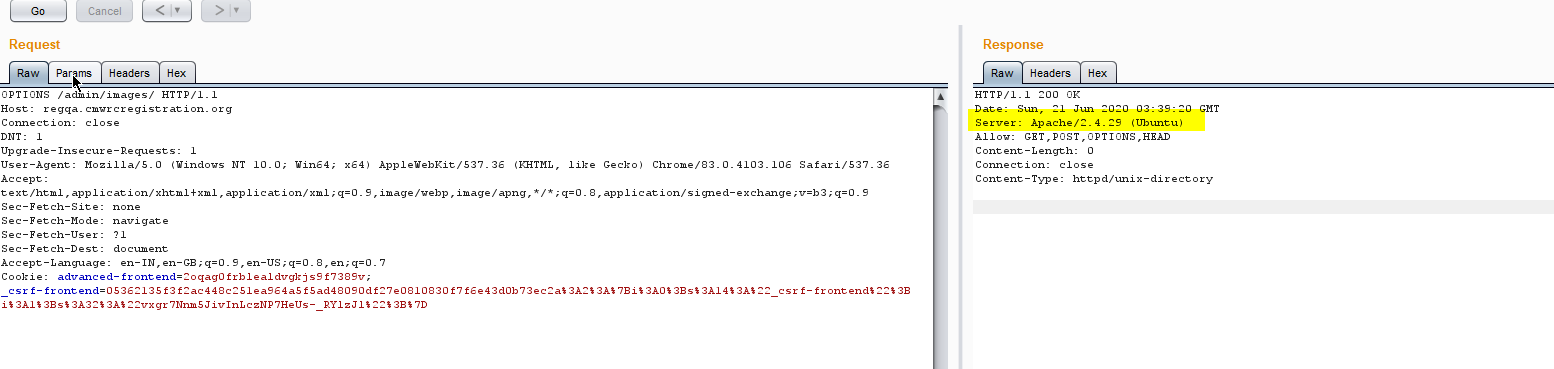
## Information Disclosure

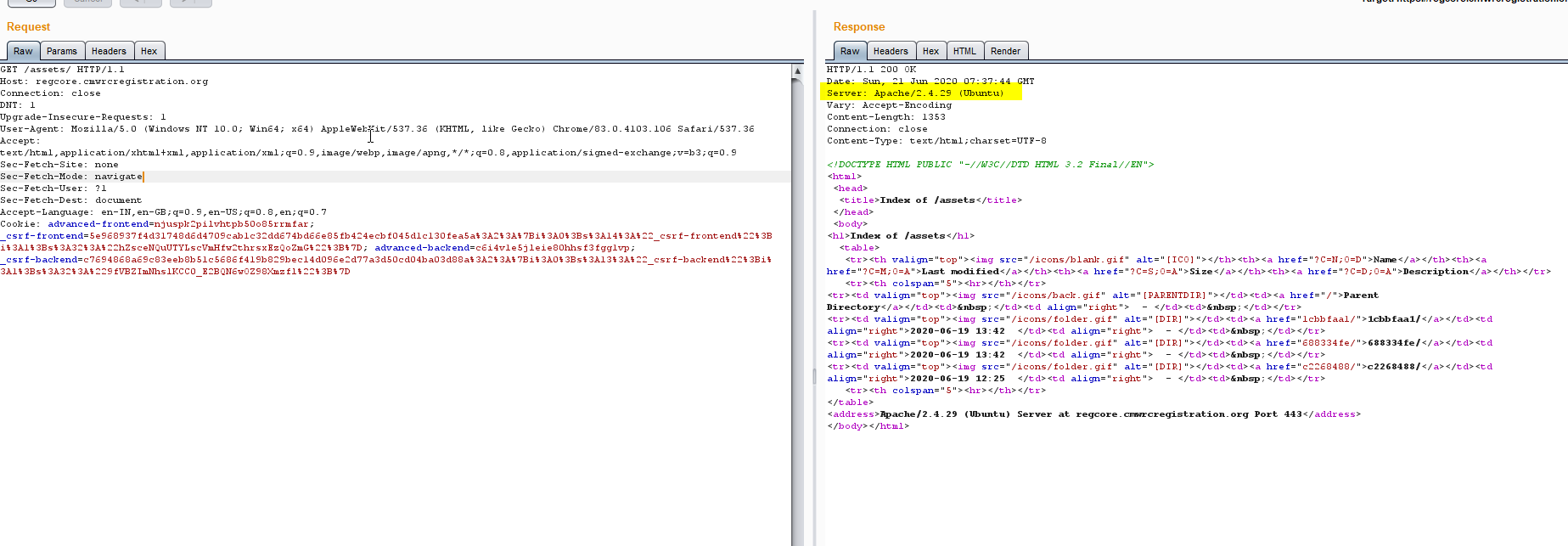
**Description**

Information disclosure is when an application fails to properly protect sensitive and confidential information from parties that are not supposed to have access to the subject matter in normal circumstances.

Returns the server information’s.

**Screenshots**





**Steps to reproduce**

* Log in to the applications
* Capture the server response using Burp suite proxy tools
* Response returns the server information’s

## Frameable response (potential Clickjacking)

**Description**

Clickjacking is an interface-based attack in which a user is tricked into clicking on actionable content on a hidden website by clicking on some other content in a decoy website

**Remediation**

We have discussed a commonly encountered browser-side prevention mechanism, namely frame busting scripts. However, we have seen that it is often straightforward for an attacker to circumvent these protections. Consequently, server driven protocols have been devised that constrain browser iframe usage and mitigate against clickjacking.

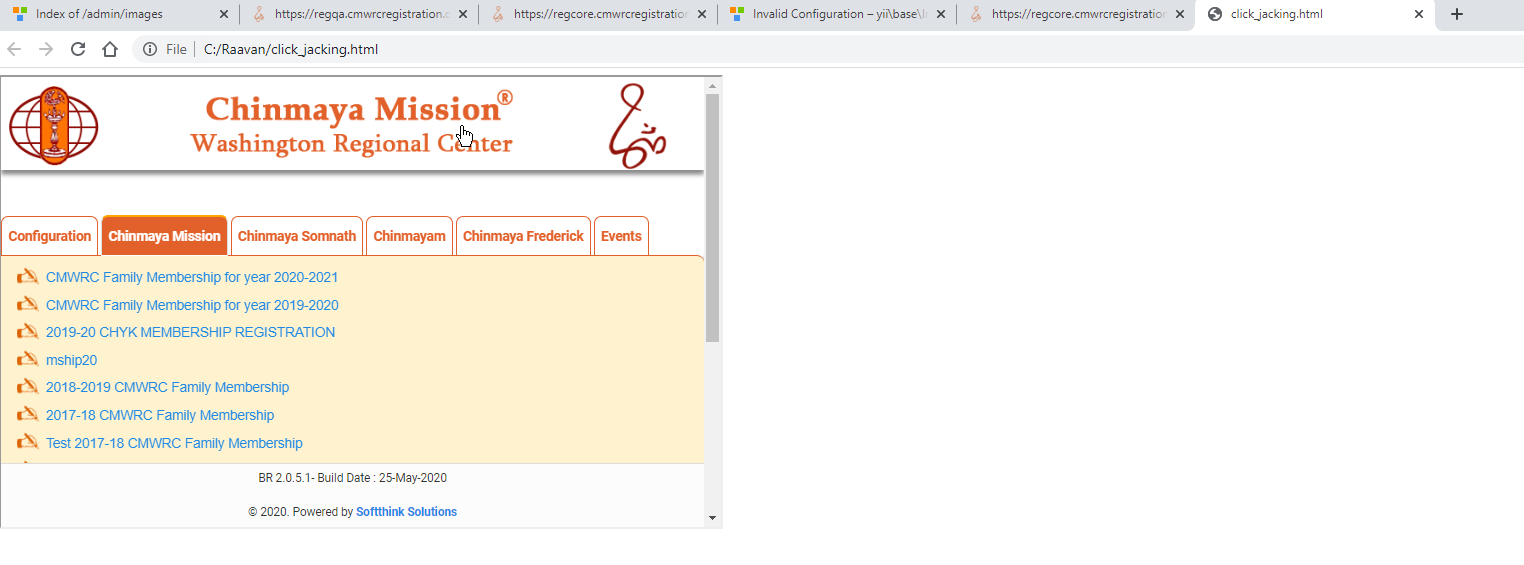
Clickjacking is a browser-side behavior and its success or otherwise depends upon browser functionality and conformity to prevailing web standards and best practice. Server-side protection against clickjacking is provided by defining and communicating constraints over the use of components such as iframes. However, implementation of protection depends upon browser compliance and enforcement of these constraints. Two mechanisms for server-side clickjacking protection are X-Frame-Options and Content Security Policy.

**Path**:

* <https://regqa.cmwrcregistration.org/site/login>
* <https://regcore.cmwrcregistration.org/site/login>

**Screenshots**







## Application Exception

**Description**

Application Exceptions are vulnerabilities where unexpected inputs can trigger inappropriate exceptions, or error responses disclosing implementation information, such as a stack trace. The attack sends various unusual inputs and looks for text in responses evidencing poor error handling.

**Remediation**

Ensure that the application input parameters are properly validated and checked for unexpected input, including null or empty values. Also, ensure that your application architecture for handling error conditions is securely implemented.

* We recommend a standard approach to application exception handling across the application (i.e. implement a standard application exception page that is displayed when errors occur).
* Return only generic error information to the client, not specific paths, directories, or the names of application components.
* Minimize exposure by limiting the type of information that is disclosed when an error occurs, so that IP addresses, usernames, or passwords are not displayed when an error occurs.

**Path:**

<https://regqa.cmwrcregistration.org/admin/site/login>

<https://regcore.cmwrcregistration.org/admin/site/login>

**Screenshots**

