Web Browser XSS Protection

Steps:

1. **httpd.conf** file was modified as follows under #Main Server Configuration:

   <IfModule mod_headers.c>
   Header set X-XSS-Protection "1; mode=block"
   </IfModule>

   Note:

   1
   Enables XSS filtering (usually default in browsers). If a cross-site scripting attack is detected, the browser will sanitize the page (remove the unsafe parts).

   1; mode=block
   Enables XSS filtering. Rather than sanitizing the page,the browser will prevent rendering of the page if an attack is detected.

2. Integration of Content Security Policy (CSP)

   Add the following to your **httpd.conf file** in the **<IfModule mod_headers.c>**:

   Header set Content-Security-Policy "script-src 'self' www.google.com/recaptcha/api.js www.gstatic.com/recaptcha/api2/v1565591531251/recaptcha__en.js 'unsafe-inline' 'unsafe-eval'; object-src 'self'; "

   It must look like the image below:

```
#XSS protection
<IfModule mod_headers.c>
  Header set X-XSS-Protection "1; mode=block"
  Header set Content-Security-Policy "script-src 'self' www.google.com/recaptcha/api.js www.gstatic.com/recaptcha/api2/v1565591531251/recaptcha__en.js 'unsafe-inline' 'unsafe-eval'; object-src 'self'; "
</IfModule>
```

Note:

The https://www.google.com/recaptcha/api.js cannot be downloaded as the line po.src contain a timestamp which determines the version in which it was deployed.

You are getting a version of api.js based on your physical location.

If you **download** this code, it means:

1. You will be using a frozen codebase, and therefore not get any benefit of any changes Google might make.
2. You are forcing users worldwide to use a version meant for a particular locale. For example: Your Japanese users will have problems with captchas designed for Italians.
3. If Google decides to remove that particular revision, your code will completely break

script-src: This defines the sources from where scripts may be loaded.

script-src 'self' www.google.com/recaptcha/api.js www.gstatic.com/recaptcha/api2/v1565591531251/recaptcha__en.js - Defines valid sources of JavaScript.

'unsafe-eval' - Allows unsafe dynamic code evaluation such as JavaScript eval().

'unsafe-inline' - Allows use of inline source elements such as style attribute, onclick, or script tag bodies

object-src 'self' - Defines valid sources of plugins, eg <object>, <embed> or <applet>

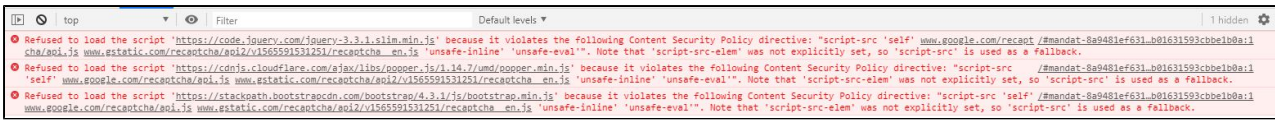
These 3 attributes was used as we had many inline styles in the application as well as inline scripts.

3. Test in browser:

For example: Suppose we want to add bootstrap scripts to our code through cdn as shown below:

```
<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.7/umd/popper.min.js" integrity="sha384-UO2eT0CpHqdSJQ6hJty5KVphtPhzWj9WO1clHTMGa3JDZwrnQq4sF86dIHNDz0W1" crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js" integrity="sha384-JjSmVgyd0p3pXB1rRibZUAYoIIy6OrQ6VrjIEaFf/nJGzIxFDsf4x0xIM+B07jRM" crossorigin="anonymous"></script>
```

Our browser will refuse to load the script as it violates our CSP policy.



Therefore, we are safe from any scripts coming from any external resource.

Reference: https://content-security-policy.com/, https://blog.cloudpassage.com/2018/02/15/xss-risk-with-apache-content/