**Content Security Policy (CSP)**

**Content Security Policy** ([CSP](https://developer.mozilla.org/en-US/docs/Glossary/CSP)) is an added layer of security that helps to detect and mitigate certain types of attacks, including Cross-Site Scripting ([XSS](https://developer.mozilla.org/en-US/docs/Glossary/Cross-site_scripting)) and data injection attacks. These attacks are used for everything from data theft, to site defacement, to malware distribution.

To enable CSP, you need to configure your web server to return the [Content-Security-Policy](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy) HTTP header. (Sometimes you may see mentions of the X-Content-Security-Policy header, but that's an older version and you don't need to specify it anymore.)

**Directives:**

CSP Directives are Categorized as

1. Fetch Directives
2. Document Directives
3. Navigation Directives
4. Reporting Directives
5. Other Directives

In our app CSP: **C:\anu\csp**

We used express generator to create the folder structure

In www we createserver1 using Https Module

And the server listens at port 8000

<https://localhost:8000>

inside our views directory and in partials we have a file nav.ejs

C:\anu\csp\views\partials\nav.ejs

Inside the nav.ejs we have navbars and five main directives

1. Fetch Directives
2. Document Directives
3. Navigation Directives
4. Reporting Directives
5. Other Directives

Inside the directives headings we have respective directives like child src and so on

And in Partials we have header and footer.ejs which holds header and footer information’s respectively.

In violationevent.ejs all the violations are stored inside the table that will be included all the directives ejs files

csp\views\partials\header.ejs

csp\views\partials\footer.ejs

csp\views\partials\.violationevent.ejs

In our main file app.js we have created six routers **index.js, fetchdir.js**,

**Docdir.js, navdir.js docdir.js** and **otherdir.js**. which will render ejs files according to their mode and header

We are using Three **modes CSP-none, CSP-enforced and CSP-report**

**FETCH DIRECTIVES:**

**Child-src:**

**<a href="/fetchdir/childsrc?mode=csp-none" class="nav- link">csp-none</a>**

**var mode = req. query["mode"];**

we get the mode using req. query method and store it inside a variable mode and based on the condition it will render the childsrc.ejs

if (mode=” CSP-none”)

it will render the page without csp header

elseif (mode==” CSP-enforced”)

res. set Header (“Content-Security-Policy”, “child-src ‘none’”);

In the enforced mode it will block the resources and throw violations

elseif (mode==” CSP-report”)

res. set Header (“Content-Security-Policy-Report-Only”, “child-src ‘none’; reporturi **https://localhost/8000/reporturi**”);

In report only mode it will not block the resources but only display the violation event

In the same manner all the directives are checked and based on the mode they will render the pages respectively.

**child-src** directive defines the valid sources for nested browsing contexts loaded using elements such as **frame** and **I frame**.

**csp\public\sourcefile.html**

In our example we load the I frame inside the src attribute value we specify the sourcefile.html which will be blocked when we specify the header as **child-src none**

[**csp\views\fetchdir\childsrc.ejs**](file:///C:\anu\csp\views\fetchdir\childsrc.ejs)

<html>

<frame width="300px" height="300px" src="/sourcefile.html" >

</frame >

</html>

sourcefile.html

==============

<html>

<body>

<h4> source file</h4>

<p>This is Source file</p>

<p id="cont">

<script>

document.getElementById("cont").innerHTML="Content from Internalscript";

</script>

</body>

</html>

**Connect-Src:**

**connect-src** directive restricts the URLs which can be loaded using script interfaces.

 It will connect to the server endpoint by using the syntax and display the data from the server without csp header.

**xhr. open ("GET","/fetchdir/getdata”, true)**

But if we specify Header connect-src none

It will throw the below error

Refused to connect to 'https://localhost:8000/fetchdir/getdata' because it violates the following

**Content Security Policy directive: "connect-src 'none'"**

<html>

<div id="data">

<button type="button" onclick="loadXMLDoc()">Change Content</button>

</div>

<script>

function loadXMLDoc(){

var xhr=new XMLHttpRequest();

xhr.onreadystatechange=function(){

if(this.readyState==4 && this.status==200){

document.getElementById("data").innerHTML=this.responseText;

}

}

xhr.open("GET","/fetchdir/getdata",true);

xhr.send();

}

**</script>**

**</div>**

**</html>**

**getdata**

**router.get("/getdata", function (req, res) {**

**res.send("getting data from the server");**

**});**

[**csp\views\fetchdir\connectsrc.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\connectsrc.ejs)

**default-Src:**

The **HTTP Content-Security-Policy (CSP) default-src** directive serves as a fallback for the other CSP fetch directives. For each of the directives that are absent, the user agent looks for the default-src directive and uses this value for it:

If there are other directives specified, default-src does not influence them**.**

**Content-Security-Policy", "default-src \*; img-src 'none'”**

Will be same as **font-src ‘\*’ style-src ‘\*’**

In our example we are loading an image since we have specified img-src ‘none’ it will not be allow the image to be loaded

**[csp\views\fetchdir\defaultsrc.ejs](C:\\anu\\csp\\views\\fetchdir\\defaultsrc.ejs)**

<html>

<img src="/images/cspimg.png">

</html>

**Font-Src:**

The **HTTP Content-Security-Policy (CSP)** **font-src** directive specifies

  valid sources for fonts loaded using @font-face.

@import <https://fonts.googleapis.com/css?family=Poppins:300,400,500,600,700>;

p {

    font-family: 'Poppins', sans-serif;

     }

 Since we have Specified **font-src 'self'** in our CSP header

  It will be refused to load the font from external site like google

**[csp\views\fetchdir\fontsrc.ejs](C:\\Users\\darsh\\AppData\\Local\\Packages\\Microsoft.Office.Desktop_8wekyb3d8bbwe\\LocalCache\\Roaming\\Microsoft\\Word\\csp\\views\\fetchdir\\fontsrc.ejs)**

<html>

<iframe width="300px" height="300px" src="/cspdir.html"></iframe>

</html>

**frame-Src:**

**HTTP**[**Content-Security-Policy**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy) **(CSP)** **frame-src** directive specifies valid sources for nested browsing contexts loading using elements such as [<frame>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/frame) and [<iframe>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe).

The img is blocked and won't load:

 Since we have specified **frame-src none** in our CSP header

[csp\views\fetchdir\framesrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\framesrc.ejs)

<html>

<iframe width="300px" height="300px" src="/cspdir.html"></iframe>

</html>

cspdir.html

<html>

<body>

<h1> Csp Directives</h1>

</body>

</html>

**img-Src:**

The **HTTP Content-Security-Policy img-src** directive specifies valid sources of images and favicons.

The img is blocked and won't load:

 Since we have specified **img-src none** in our CSP header

[csp\views\fetchdir\imgsrc.ejs](C:\\Users\\darsh\\AppData\\Local\\Packages\\Microsoft.Office.Desktop_8wekyb3d8bbwe\\LocalCache\\Roaming\\Microsoft\\Word\\csp\\views\\fetchdir\\imgsrc.ejs)

<html>

<img src="/images/cspimg.png" width="300px" height="300px">

</html>

**Manifest-Src:**

**The HTTP Content-Security-Policy: manifest-src** directive specifies which manifest can be applied to the resource.

The following link is blocked and won't load:

**link rel="manifest" href="/manifest.json"**

Because we have specified **manifest-src none**

[csp\views\fetchdir\manifestsrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\manifestsrc.ejs)

**Browser Support: Chrome, Edge, Opera,Safari**

<html>

<video width="300px" height="300px" controls>

<source src="/media/sample-mp4-file-small.mp4" type="video/mp4">

</video>

</html>

**Media-Src:**

**The HTTP Content-Security-Policy (CSP) media-src** directive specifies valid sources for loading media using the <audio> and <video> elements.

**Content-Security-Policy media-src 'none'**

The video elements are blocked and won't load: since we have specified media-src none</p>

[csp\views\fetchdir\mediasrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\mediasrc.ejs)

<html>

<video width="300px" height="300px" controls>

<source src="/media/sample-mp4-file-small.mp4" type="video/mp4">

</video>

</html>

**Object-Src:**

The **HTTP Content-Security-Policy object-src** directive specifies valid sources for the object embed, and applet elements**.**

**<object data="/cspdir.html" width="800px" height="300px"></object>**

**Content-Security-Policy object-src 'none'**

The object elements are blocked and won't load: since we have specified object-src none

[csp\views\fetchdir\objectsrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\objectsrc.ejs)

<html>

<object data="/cspdir.html" width="800px" height="300px"></object>

</html>

**Prefetch-src:**

**The HTTP Content-Security-Policy (CSP) prefetch-src directive** specifies valid resources that may be prefetched or prerendered**.**

**<link rel="preload" as="script"    href="/javascripts/scriptfile.js">**

**Content-Security-Policy prefetch-src 'none'**

It will throw the below error

Refused to prefetch content from **'https://localhost:8000/stylesheets/prefetchstyle.css**'

because it violates the following Content Security Policy directive: "prefetch-src 'none'".

**Browser Support: None of the browser supports**

[csp\views\fetchdir\prefetchsrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\prefetchsrc.ejs)

<html>

<a href="/prefetch.html">prefetchfile</a>

</html>

**Script-src**

**The HTTP Content-Security-Policy (CSP)** script-src directive specifies valid sources for JavaScript. This includes not only URLs loaded directly into <script> elements, but also things like inline script event handlers (onclick) which can trigger script execution.

we have --external script and inline script trigger script as well as svg onload="alert (1)" script attributes

if we specify our header like this

**Content-Security-Policy", "script-src 'self'**

It will block the external script, inline script as well as inline handler also

**Browser Support: Chrome, Edge, Firefox ,Opera**

[csp\views\fetchdir\scriptsrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\scriptsrc.ejs)

<html>

<p id="demo"></p>

<!--External script-->

<p>Text from External Script</p>

<script src="/javascripts/scriptfile.js"></script>

<!--button trigger the inline script-->

<p id="inline"></p>

<svg onload="alert(1)"></svg>

<!-- Script attribute-->

</html>

<script>

document.getElementById("inline") .innerHTML=" Text From Inline Script"

document.getElementById("demo").innerHTML="This is content from Inline Script"

</script>

**Script-src-attr:**

The HTTP Content-Security-Policy (CSP) script-src-attr directive specifies valid sources for JavaScript inline event handlers. This includes only inline script event handlers like onclick, but not URLs loaded directly into script elements.

**"Content-Security-Policy", " script-src-attr 'self**

[C:\anu\csp\views\fetchdir\scriptsrc.ejs](file:///C:\anu\csp\views\fetchdir\scriptsrc.ejs)

**Browser Support: Chrome, Edge, Opera**

**Not supported by Firefox**

**Script-src-elem:**

The HTTP Content-Security-Policy (CSP) script-src-elem directive specifies valid sources for JavaScript script elements, but not inline script event handlers like onclick.

**"Content-Security-Policy-Report-Only", " script-src-elem 'self**

[C:\anu\csp\views\fetchdir\scriptsrc.ejs](file:///C:\anu\csp\views\fetchdir\scriptsrc.ejs)

**Browser Support: Chrome, Edge, Opera**

**Not supported by Firefox and Safari**

**Style-src:**

**The HTTP Content-Security-Policy (CSP)** style-src directive specifies valid sources for stylesheets.

The following stylesheets are blocked and won't load: since we have specified style-src ‘none’

We have external stylesheet which will change the font

<link rel="stylesheet" href="/stylesheets/style.css">

And internal style tag which will change the body color

<style>background-color: red; </style>

And inline style which will change the font color

<p style=” color: blue”></p>

[C:\anu\csp\views\fetchdir\stylesrc.ejs](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\stylesrc.ejs)

<html>

<p style="color:blue;"> style-src:CSP</p>

</html>

**Browser Support: Chrome, Edge, Firefox, Opera,Safari**

**Style-src-attr:**

The HTTP [Content-Security-Policy](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy) (CSP) **style-src-attr** directive specifies valid sources for inline styles applied to individual DOM elements.

[csp\views\fetchdir\stylesrc.ejs](file:///C:\anu\csp\views\fetchdir\stylesrc.ejs)

**Browser Support: Chrome, Edge, Opera**

**Not supported by Firefox and Safari**

**Style-src-elem:**

The HTTP [Content-Security-Policy](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy) (CSP) **style-src-elem** directive specifies valid sources for stylesheets [<style>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/style) elements and [<link>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/link) elements with rel="stylesheet".

[csp\views\fetchdir\stylesrc.ejs](file:///C:\anu\csp\views\fetchdir\stylesrc.ejs)

**Browser Support: Chrome, Edge, Opera**

**Not supported by Firefox and Safari**

**Worker-src:**

The HTTP Content-Security-Policy (CSP) worker-src directive specifies valid sources for Worker, SharedWorker, or ServiceWorker scripts.

Inside the Main.js file we are calling workers and worker will post the message I am worker file

Main js file

**Content-Security-Policy", "worker-src 'none'**

[**csp\views\fetchdir\workersrc.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\fetchdir\workersrc.ejs)

**<html>**

**<p id="res"></p>**

**<script src="/javascripts/main.js"></script>**

**</html>**

**document.write("Main js file");**

**var w=new Worker("/javascripts/workers.js");**

**w.onmessage=function(event)**

**{**

**document.getElementById("res").innerHTML=event.data;**

**}**

**workers.js**

**postMessage("I am worker file");**

**Browser Support: Chrome, Firefox, Edge, Opera**

The Worker is blocked and won't load: since we have specified object-src none

**FETCH DIRECTIVES:**

**Base Uri:**

**The HTTP**[**Content-Security-Policy**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy)**base-uri** directive restricts the URLs which can be used in a document's [<base>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/base) element.

<base href="/images/">

<img src="cspimg.png" width="300px" height="300px">

We are loading the image in the src tag we specify the relative path so it will get the image file form the images directory since we have specified the path in base tag.

If we Set CSP Header like below

**Content-Security-Policy", "base-uri 'none'**

**[C:\anu\csp\views\docdir\baseuri.ejs](C:\\Users\\darsh\\AppData\\Local\\Packages\\Microsoft.Office.Desktop_8wekyb3d8bbwe\\LocalCache\\Roaming\\Microsoft\\Word\\csp\\views\\docdir\\baseuri.ejs)**

It will not load the image

because it violates the following Content Security Policy directive: "base-uri 'none'".

**Browser Support: Chrome, Edge, Opera**

**Sandbox:**

The **HTTP**[**Content-Security-Policy**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy)**(CSP**) sandbox directive enables a sandbox for the requested resource similar to the [<frame>](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe) [sandbox](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe#attr-sandbox) attribute.

Inside the frame we are trying to submit the form after clicking the submit button the form submission will not happen if we specify

**Content-Security-Policy sandbox**

If we want allow the form to be submitted inside the I frame then we have to specify

**Content-Security-Policy", "sandbox allow-forms**

since **sandbox** will not support **report-only** we have to specify **sandbox** as empty in enforced mode and sandbox with attribute values like allow-scripts allow-forms to allow the resources to load in report mode

[**csp\views\docdir\sandbox.ejs**](file:///C:\anu\csp\views\docdir\sandbox.ejs)

**<html>**

**<iframe id="frameobj" src="/submitform.html"></iframe>**

**</html>**

**Browser Support: Chrome, Edge, Firefox, Opera,Safari**

**Navigating DIRECTIVES:**

**Form-action:**

The **HTTP** [**Content-Security-Policy**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy) **(CSP) form-action directive** restricts the URLs which can be used as the target of form submissions from a given context.

If we specify the CSP header like this

**Content Security Policy directive: "form-action 'none'".**

Error: Refused to send form data because it violates the following

**Content Security Policy directive: "form-action 'none'".**

[**csp\views\navdir\formactions.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\navdir\formactions.ejs)

<form action="JavaScript: alert ('Form action')" id="form1" method="post">

<input type="text" name="textname" value="value">

<input type="submit" id="submit" value="submit">

</form>

**Browser Support: Chrome, Edge, Firefox, Opéra**

**Frame-ancestors:**

The **HTTP**[**Content-Security-Policy**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy)**(CSP) frame-ancestors**directive specifies valid parents that may embed a pageusing[**<frame>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/frame)**,**[**<iframe>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/iframe)**,**[**<object>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/object)**,**[**<embed>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/embed)**, or**[**<applet>**](https://developer.mozilla.org/en-US/docs/Web/HTML/Element/applet)**.**

Setting this directive to 'none' is similarto[**X-Frame-Options**](https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options)**:** deny

In our example we are trying t[o load a page inside our I frame](C:\\Users\\darsh\\AppData\\Local\\Packages\\Microsoft.Office.Desktop_8wekyb3d8bbwe\\LocalCache\\Roaming\\Microsoft\\Word\\o load a page inside our I framehttps:\\localhost:8000\\public\\frame)

[https://localhost:8000/public/frame](C:\\Users\\darsh\\AppData\\Local\\Packages\\Microsoft.Office.Desktop_8wekyb3d8bbwe\\LocalCache\\Roaming\\Microsoft\\Word\\o load a page inside our I framehttps:\\localhost:8000\\public\\frame)

If we specify the CSP header like this

**Content Security Policy directive: "frame-ancestors 'none'".**

Refused to frame 'http://localhost:3000/frame' because an ancestor violates the following **Content Security Policy directive: "frame-ancestors 'none'"**

[**csp\views\navdir\frameancestors.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\navdir\frameancestors.ejs)

<html>

<iframe width="600px" height="200px" src="/navdir/frame" style="border:none">

</html>

**Navigate-to:**

**The HTTP Content-Security-Policy (CSP) navigate-to directive** restricts the URLs to which a document can initiate navigations by any means including <form> (if form-action is not specified), <a>, window. location, window. Open, etc.

                This is an enforcement on what navigations this document initiates, not on what this document is allowed to navigate to.

**Content Security Policy directive: "navigate-to 'none'".**

[**csp\views\navdir\navigateto.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\navdir\navigateto.ejs)

**<html>**

**<form action="javascript: alert('Form actions')" id="form1" method="post">**

**<input type="text" name="textName" value="textValue">**

**<input type="submit" id="submit" value="submit">**

**</form>**

**</html>**

**Browser Support: None of the browser supports**

**Reporting DIRECTIVES:**

**Report-uri:**

The deprecated **HTTP Content-Security-Policy (CSP) report-uri** directive instructs the user agent to report attempts to violate the Content Security Policy. These violation reports consist of JSON documents sent via an **HTTP POST** request to the specified URI.

**Content Security Policy directive: "form-action 'none’; report-uri: https://loacalhost:8000/reporturi".**

It will send the violation report to the specified URL

[**https://loacalhost:8000/reporturi**](https://loacalhost:8000/reporturi)**".**

[**csp\views\reportdir\reporturi.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\reportdir\reporturi.ejs)

**Browser Support: Chrome, Edge, Firefox, Opéra**

<html>

<form action="javascript:alert('Form actions')" id="form1" method="post">

<input type="text" name="textName" value="textValue">

<input type="submit" id="submit" value="submit">

</form>

</html>

**Report- to:**

The **Content-Security-Policy *Report-To* HTTP** response header field instructs the user agent to store reporting endpoints for an origin.

**Content-Security-Policy: ...; report-to groupname**

const reportCSP=

        {"group": "csp-endpoint",

        "Max\_age": "10000000",

        "endpoints": [{"url":"https://localhost:8000/reporturi"}]

          };

var header = "Content-Security-Policy-Report-Only form-action 'none' ";

 res. setHeader ("report-to", JSON.stringify(reportCSP));

reportCSP is the group name and maxage specify the expiration in hours

url should be https not http, and you should specify both report-uri report-to in the header

**Content-Security-Policy-Report-Only", "form-action 'none’; report-uri https://localhost:8000/reporturi report-to default**

Then only it will be supported by the browser

[**csp\views\reportdir\reportto.ejs**](file:///C:\Users\darsh\AppData\Local\Packages\Microsoft.Office.Desktop_8wekyb3d8bbwe\LocalCache\Roaming\Microsoft\Word\csp\views\reportdir\reportto.ejs)

**<html>**

**<form action="javascript:alert('Form actions')" id="form1" method="post">**

**<input type="text" name="textName" value="textValue">**

**<input type="submit" id="submit" value="submit">**

**</form>**

**</html>**

**Browser Support: Chrome, Edge**

**For browser Compatibility check the below link**

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP>