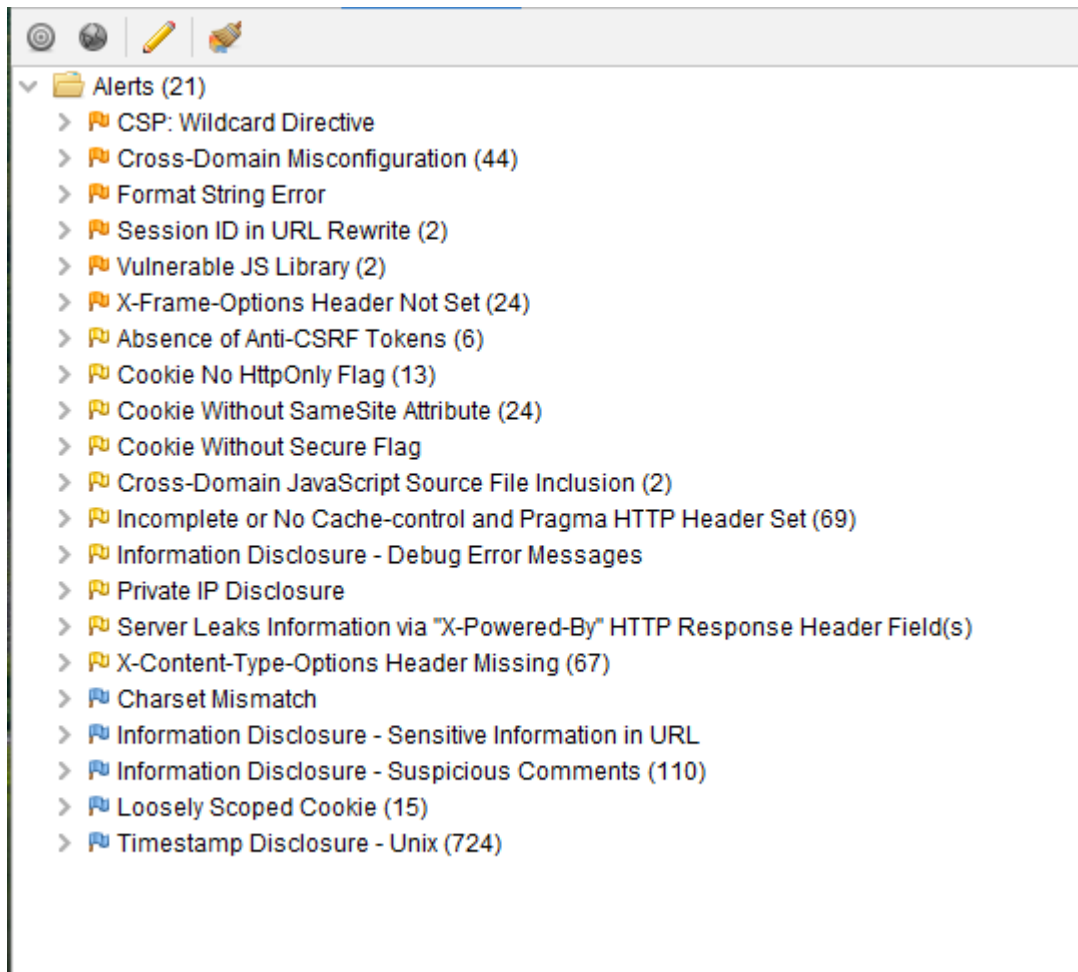


## Zap Testing & Risk Mitigation

### Summary of Alerts after running an active scan with ZAP:



Scanning with ZAP did not discover any high security vulnerabilities, but it did reveal a bunch of medium and low-level security risks.

I focused on 3 separate medium risk vulnerabilities that are detailed below along with the steps I took to mitigate them.

## Medium Vulnerability #1:

### CSP: Wildcard Directive

<div>Alerts (21)</div> <div>CSP: Wildcard Directive</div> <div>Cross-Domain Misconfiguration (44)</div> <div>Format String Error</div> <div>Session ID in URL Rewrite (2)</div> <div>Vulnerable JS Library (2)</div> <div>X-Frame-Options Header Not Set (24)</div> <div>Absence of Anti-CSRF Tokens (6)</div> <div>Cookie No HttpOnly Flag (13)</div> <div>Cookie Without SameSite Attribute (24)</div> <div>Cookie Without Secure Flag</div> <div>Cross-Domain JavaScript Source File Inclusion (2)</div> <div>Incomplete or No Cache-control and Pragma HTTP Header Set (69)</div> <div>Information Disclosure - Debug Error Messages</div> <div>Private IP Disclosure</div> <div>Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)</div> <div>X-Content-Type-Options Header Missing (67)</div> <div>Charset Mismatch</div> <div>Information Disclosure - Sensitive Information in URL</div> <div>Information Disclosure - Suspicious Comments (110)</div> <div>Loosely Scoped Cookie (15)</div> <div>Timestamp Disclosure - Unix (724)</div>	<div>URL: <a href="https://www.naipeua.com/minimumpatnry.com">https://www.naipeua.com/minimumpatnry.com</a></div> <div>Risk:  Medium</div> <div>Confidence: Medium</div> <div>Parameter: Content-Security-Policy</div> <div>Attack:</div> <div>Evidence: frame-ancestors 'self'</div> <div>CWE ID: 16</div> <div>WASC ID: 15</div> <div>Source: Passive (10055 - CSP)</div> <div>Description: The following directives either allow wildcard sources (or ancestors), are not defined, or are overly broadly defined: script-src, script-src-elem, script-src-attr, style-src, style-src-elem, style-src-attr, img-src, connect-src, frame-src, font-src, media-src, object-src, manifest-src, worker-src, prefetch-src, form-action</div> <div>Other Info:</div> <div>Solution: Ensure that your web server, application server, load balancer, etc. is properly configured to set the Content-Security-Policy header.</div> <div>Reference: <a href="http://www.w3.org/TR/CSP2/">http://www.w3.org/TR/CSP2/</a> <a href="http://www.w3.org/TR/CSP/">http://www.w3.org/TR/CSP/</a> <a href="http://caniuse.com/#search=content+security+policy">http://caniuse.com/#search=content+security+policy</a></div>
--	--

### Mitigation:

To implement the Content-Security-Policy, I created the *web.config* file and added the following line of code to add the following explicit policy directives. This helps mitigate Cross Site Scripting (XCC) and data injection attacks.

```
<httpProtocol>
  <customHeaders>
    <add name="Content-Security-Policy" value="upgrade-insecure-requests; default-src 'self'; style-src 'self'; worker-src 'self';
    form-action 'self'; prefetch-src 'self'; frame-src 'self'; manifest-src 'self'; connect-src 'self';
    script-src 'self'; object-src 'none'; media-src 'none'; font-src 'none'; img-src 'none'; frame-ancestors 'self'" />
  </customHeaders>
</httpProtocol>
```

## Medium Vulnerability #2:

### Cross-Domain Misconfiguration

<div>Alerts (21)</div> <div>CSP: Wildcard Directive</div> <div>Cross-Domain Misconfiguration (44)</div> <div>Format String Error</div> <div>Session ID in URL Rewrite (2)</div> <div>Vulnerable JS Library (2)</div> <div>X-Frame-Options Header Not Set (24)</div> <div>Absence of Anti-CSRF Tokens (6)</div> <div>Cookie No HttpOnly Flag (13)</div> <div>Cookie Without SameSite Attribute (24)</div> <div>Cookie Without Secure Flag</div> <div>Cross-Domain JavaScript Source File Inclusion (2)</div> <div>Incomplete or No Cache-control and Pragma HTTP Header Set (69)</div> <div>Information Disclosure - Debug Error Messages</div> <div>Private IP Disclosure</div> <div>Server Leaks Information via "X-Powered-By" HTTP Response Header Field(s)</div> <div>X-Content-Type-Options Header Missing (67)</div> <div>Charset Mismatch</div> <div>Information Disclosure - Sensitive Information in URL</div> <div>Information Disclosure - Suspicious Comments (110)</div> <div>Loosely Scoped Cookie (15)</div> <div>Timestamp Disclosure - Unix (724)</div>	<div>Cross-Domain Misconfiguration</div> <div>URL: <a href="https://use.fontawesome.com/releases/v5.8.1/css/all.css">https://use.fontawesome.com/releases/v5.8.1/css/all.css</a></div> <div>Risk:  Medium</div> <div>Confidence: Medium</div> <div>Parameter:</div> <div>Attack:</div> <div>Evidence: Access-Control-Allow-Origin: *</div> <div>CWE ID: 264</div> <div>WASC ID: 14</div> <div>Source: Passive (10098 - Cross-Domain Misconfiguration)</div> <div>Description: Web browser data loading may be possible, due to a Cross Origin Resource Sharing (CORS) misconfiguration on the web server</div> <div>Other Info: The CORS misconfiguration on the web server permits cross-domain read requests from arbitrary third party domains, using unauthenticated APIs on this domain. Web browser implementations do not permit arbitrary third parties to read the response from authenticated APIs, however. This reduces the risk somewhat. This misconfiguration could be used by an attacker to access data that is available in an</div> <div>Solution: Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance). Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.</div> <div>Reference: <a href="http://www.hpenterprisesecurity.com/vulncat/en/vulncat/vb/html5_overly_permissive_cors_policy.html">http://www.hpenterprisesecurity.com/vulncat/en/vulncat/vb/html5_overly_permissive_cors_policy.html</a></div>
--	--

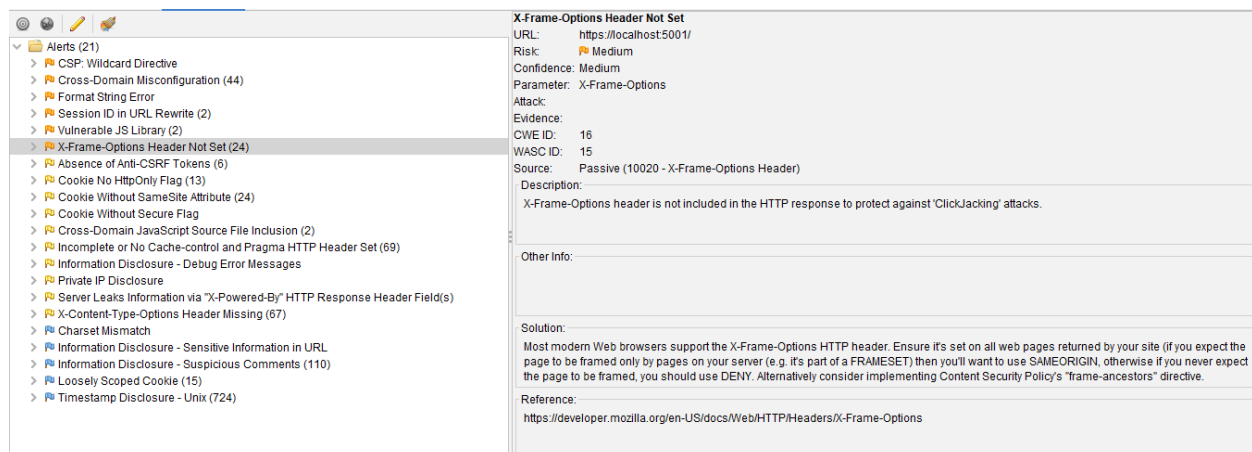
## Mitigation:

To address this specific vulnerability, I added the following line of code to the *web.config*'s *customHeaders* section. This specifies that the response header can be shared with the requesting code from the app's domain, which prevents any third-party origins from accessing the resource.

```
<add name="Access-Control-Allow-Origin" value="domain" />
</customHeaders>
```

## Medium Vulnerability #3:

### *X-Frame-Options Header Not Set*



The screenshot displays a security scanner interface. On the left, a tree view under 'Alerts (21)' lists various vulnerabilities, with 'X-Frame-Options Header Not Set (24)' selected. On the right, the details for this vulnerability are shown:

- URL:** https://localhost5001/
- Risk:** Medium
- Confidence:** Medium
- Parameter:** X-Frame-Options
- Attack:**
- Evidence:**
  - CWE ID: 16
  - WASC ID: 15
  - Source: Passive (10020 - X-Frame-Options Header)
- Description:** X-Frame-Options header is not included in the HTTP response to protect against 'ClickJacking' attacks.
- Other Info:**
- Solution:** Most modern Web browsers support the X-Frame-Options HTTP header. Ensure it's set on all web pages returned by your site (if you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.
- Reference:** <https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options>

## Mitigation:

To mitigate this vulnerability, I added the following line in the *Startup.cs* file, as well as the *X-Frame-Options* line and a *frame-ancestors* directive inside the *web.config* file. These steps help to mitigate the likelihood of clickjacking attacks against the website.

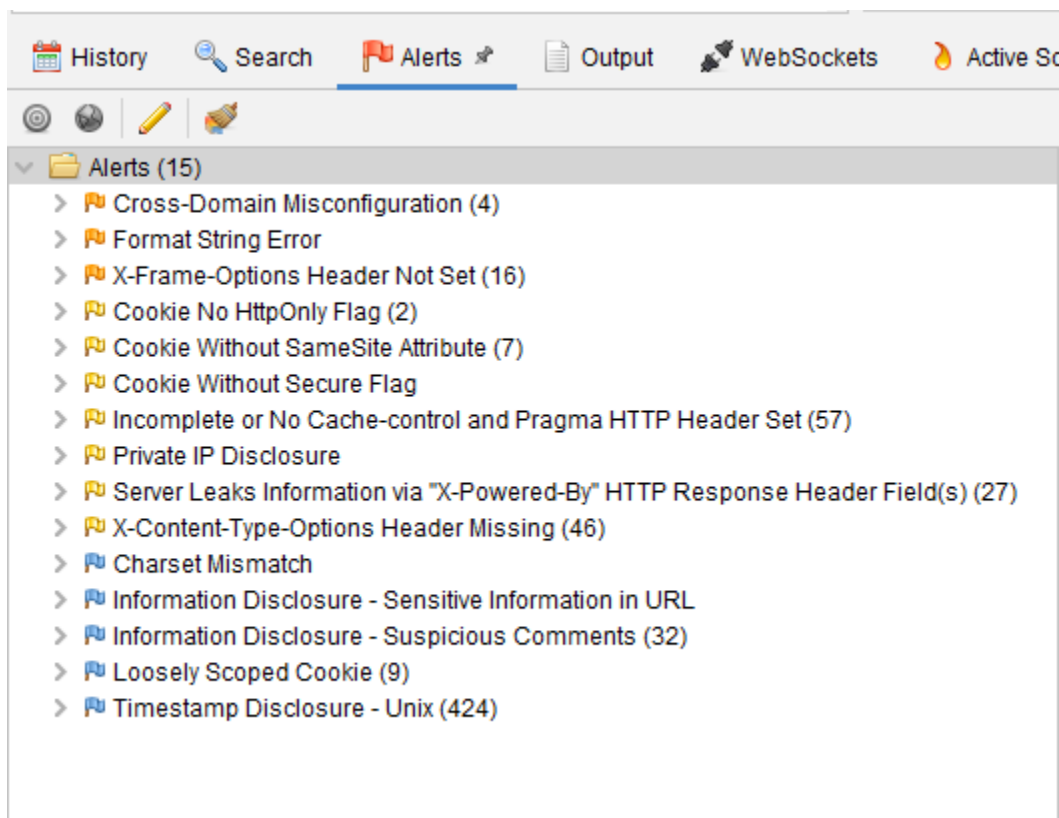
```
// mitigating X-Frame-Options exploitation
services.AddAntiforgery(options =>
{
    options.SuppressXFrameOptionsHeader = true;
});
```

```

<system.webServer>
  <httpProtocol>
    <customHeaders>
      <add name="X-Frame-Options" value="DENY" />
      <add name="Content-Security-Policy" value="upgrade-insecure-requests; default-src 'self'; style-src 'self'; worker-src 'self';
        form-action 'self'; prefetch-src 'self'; frame-src 'self'; manifest-src 'self'; connect-src 'self';
        script-src 'self'; object-src 'none'; media-src 'none'; font-src 'none'; img-src 'none'; frame-ancestors 'self'" />
      <add name="Access-Control-Allow-Origin" value="domain" />
    </customHeaders>
  </httpProtocol>

```

## Summary of Alerts after implementing steps to mitigate security risks:



As evident from the screenshot, medium vulnerability alert count is reduced significantly. Low risk vulnerabilities were affected (lowered) as well.