NETSPARKER SCAN REPORT SUMMARY

TARGET URL	https://miaplicacionweb.azurewebsites.net/
SCAN DATE	29/04/2017 23:21:36
REPORT DATE	01/05/2017 13:57:32
SCAN DURATION	00:23:31
NETSPARKER VERSION	4.8.1.14376-4.8.1-hf1-9a19bce

Total Requests
13065

Average Speed
9,26 req/sec.

20
Identified

9
Confirmed

0
Critical

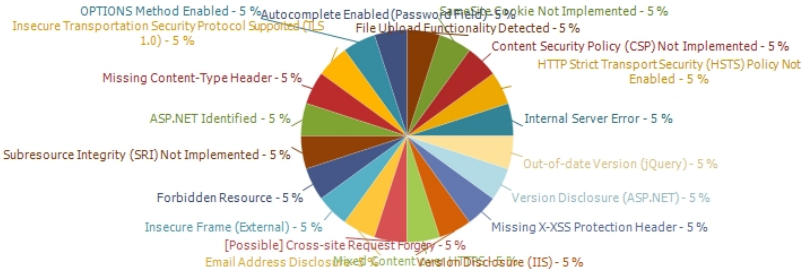
13
Informational

SCAN SETTINGS

ENABLED ENGINES	SQL Injection, SQL Injection (Boolean), SQL Injection (Blind), Cross-site Scripting, Command Injection, Command Injection (Blind), Local File Inclusion, Remote File Inclusion, Code Evaluation, HTTP Header Injection, Open Redirection, Expression Language Injection, Web App Fingerprint, RoR Code Execution, WebDAV, Reflected File Download, Insecure Reflected Content, XML External Entity, File Upload, Windows Short Filename, Server-Side Request Forgery (Pattern Based), Cross-Origin Resource Sharing (CORS), HTTP Methods, Server-Side Request Forgery (DNS), SQL Injection (Out of Band), XML External Entity (Out of Band), Cross-site Scripting (Blind), Remote File Inclusion (Out of Band), Code Evaluation (Out of Band)
URL REWRITE MODE	Heuristic
DETECTED URL REWRITE RULES	/Intercambios/Edit/{param1} /Intercambios/Details/{param1} /Intercambios/Delete/{param1} /FacturaIntercambios/Delete/{param1} /FacturaIntercambios/Edit/{param1} /FacturaIntercambios/Details/{param1}

Authentication
Scheduled

VULNERABILITIES



LOW
35 %

INFORMATION
65 %

VULNERABILITY SUMMARY

URL	Parameter	Method	Vulnerability	Confirmed
		Internal Server Error	Yes	
		Version Disclosure (ASP.NET)	No	
		Insecure Frame (External)	Yes	
		Missing Content-Type Header	No	
		Insecure Transportation Security Protocol Supported (TLS 1.0)	Yes	
		[Possible] Cross-site Request Forgery	No	
		Mixed Content over HTTPS	Yes	
		Forbidden Resource	Yes	
		File Upload Functionality Detected	Yes	
		ASP.NET Identified	No	
		Email Address Disclosure	No	
		Version Disclosure (IIS)	No	
		HTTP Strict Transport Security (HSTS) Policy Not Enabled	No	
		OPTIONS Method Enabled	Yes	
		Autocomplete Enabled (Password Field)	Yes	
		Out-of-date Version (jQuery)	No	
		Missing X-XSS Protection Header	No	
		SameSite Cookie Not Implemented	Yes	
		Subresource Integrity (SRI) Not Implemented	No	
		Content Security Policy (CSP) Not Implemented	No	

1. Internal Server Error

1 TOTAL
LOW
CONFIRMED
1

Netsparker identified an internal server error.

The server responded with an HTTP status 500, indicating there is a server-side error. Reasons may vary, and the behavior should be analyzed carefully. If Netsparker is able to find a security issue in the same resource, it will report this as a separate vulnerability.

Impact

The impact may vary depending on the condition. Generally this indicates poor coding practices, not enough error checking, sanitization and whitelisting. However, there might be a bigger issue, such as SQL injection. If that's the case, Netsparker will check for other possible issues and report them separately.

Remedy

Analyze this issue and review the application code in order to handle unexpected errors; this should be a generic practice, which does not disclose further information upon an error. All errors should be handled server-side only.

Classification

1.1. Confirmed

Request
Response

2. Version Disclosure (ASP.NET)

1 TOTAL

LOW

Netsparker identified a version disclosure (ASP.NET) in target web server's HTTP response.

This information can help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of ASP.NET.

Impact

An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.

Remedy

Apply the following changes to your `web.config` file to prevent information leakage by using custom error pages and removing `X-AspNet-Version` from HTTP responses.

```
<System.Web>
     <httpRuntime enableVersionHeader="false" />
     <customErrors mode="On" defaultRedirect="~/error/GeneralError.aspx">
          <error statusCode="403" redirect="~/error/Forbidden.aspx" />
          <error statusCode="404" redirect="~/error/PageNotFound.aspx" />
          <error statusCode="500" redirect="~/error/InternalError.aspx" />
     </customErrors>
</System.Web>
```

Remedy References

- Error Handling in ASP.NET Pages and Applications
- Remove Unwanted HTTP Response Headers

Classification

CWE-205 CAPEC-170 WASC-45 HIPAA-164.306(A), 164.308(A)

2.1.

Certainty

Request

Response

3. Insecure Frame (External)

1 TOTAL
LOW
CONFIRMED
1

Netsparker identified an external insecure or misconfigured iframe.

Impact

IFrame sandboxing enables a set of extra restrictions for the content in the inline frame.

Same Origin policy allows one window to access properties/functions of another one only if they come from the same protocol, the same port and also the same domain.

URLs from the same origin:

```
http://site.com
http://site.com/
http://site.com/my/page.html
```

URLs not from the same origin:

```
http://www.site.com  (sub domain)
http://site.org      (different domain)
https://site.com     (different protocol)
http://site.com:8080 (different port)
```

When the `sandbox` attribute is set, the iframe content is treated as being from a unique origin and sandboxed content is re-hosted in the browser with the following restrictions:

- Plugins are disabled. Any kind of ActiveX, Flash, or Silverlight plugin will not be executed.
- Forms are disabled. The hosted content is not allowed to make forms post back to any target.
- Scripts are disabled. JavaScript is disabled and will not execute.
- Links to other browsing contexts are disabled. An anchor tag targeting different browser levels will not execute.
- Unique origin treatment. All content is treated under a unique origin. The content is not able to traverse the DOM or read cookie information.

When not set or misconfigured `sandbox` or `seamless` attribute of an iframe for an untrusted URL:

- Compromised website in the iframe might affect the users in parent web application.
- Sandbox containing a value of :
  - `allow-same-origin` will not force the unique origin for iframe contents.
  - `allow-top-navigation` will allow iframe to navigate parent context, e.g. change `parent.location`.
  - `allow-forms` will allow forms submissions from inside iframe.
  - `allow-popups` will allow popups.
  - `allow-scripts` will allow malicious script execution however still disallow to create popups.
- If `seamless` attribute is set, links within the iframe will navigate the parent frame.

Remedy

- Apply sandboxing in inline frame

```
<iframe sandbox src="framed-page-url"></iframe>
```

- For untrusted content, avoid the usage of `seamless` attribute and `allow-top-navigation`, `allow-popups` and `allow-scripts` in sandbox attribute.

External References

- HTML5 Security Cheat Sheet

Remedy References

- How to Safeguard your Site with HTML5 Sandbox
- Play safely in sandboxed IFrames

Classification

3.1. Confirmed

Request

Response

4. Missing Content-Type Header

1 TOTAL

LOW

Netsparker detected a missing `Content-Type` header which means that this website could be at risk of a MIME-sniffing attacks.

Impact

MIME type sniffing is a standard functionality in browsers to find an appropriate way to render data where the HTTP headers sent by the server are either inconclusive or missing.

This allows older versions of Internet Explorer and Chrome to perform MIME-sniffing on the response body, potentially causing the response body to be interpreted and displayed as a content type other than the intended content type.

The problem arises once a website allows users to upload content which is then published on the web server. If an attacker can carry out XSS (Cross-site Scripting) attack by manipulating the content in a way to be accepted by the web application and rendered as HTML by the browser, it is possible to inject code in e.g. an image file and make the victim execute it by viewing the image.

Remedy

1. When serving resources, make sure you send the content-type header to appropriately match the type of the resource being served. For example, if you are serving an HTML page, you should send the HTTP header:

   ```
   Content-Type: text/html
   ```

2. Add the X-Content-Type-Options header with a value of "nosniff" to inform the browser to trust what the site has sent is the appropriate content-type, and to not attempt "sniffing" the real content-type.

   ```
   X-Content-Type-Options: nosniff
   ```

External References

- MIME Sniffing: feature or vulnerability?

Classification

OWASP 2013-A5

4.1.

Certainty

Request

Response

5. Insecure Transportation Security Protocol Supported (TLS 1.0)

1 TOTAL
LOW
CONFIRMED
1

Netsparker detected that insecure transportation security protocol (TLS 1.0) is supported by your web server.

TLS 1.0 has several flaws. An attacker can cause connection failures and they can trigger the use of TLS 1.0 to exploit vulnerabilities like BEAST (Browser Exploit Against SSL/TLS).

Websites using TLS 1.0 will be considered non-compliant by PCI after 30 June 2018.

Impact

Attackers can perform man-in-the-middle attacks and observe the encryption traffic between your website and its visitors.

Remedy

Configure your web server to disallow using weak ciphers. You need to restart the web server to enable changes.

- For Apache, adjust the SSLProtocol directive provided by the mod_ssl module. This directive can be set either at the server level or in a virtual host configuration.

  ```
  SSLProtocol +TLSv1.1 +TLSv1.2
  ```

- For Nginx, locate any use of the directive ssl_protocols in the `nginx.conf` file and remove `TLSv1`.

  ```
  ssl_protocols TLSv1.1 TLSv1.2;
  ```

- For Microsoft IIS, you should make some changes on the system registry.
  1. Click on Start and then Run, type `regedt32` or `regedit`, and then click OK.
  2. In Registry Editor, locate the following registry key or create if it does not exist:

     ```
     HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.0\
     ```

  3. Locate a key named `Server` or create if it doesn't exist.
  4. Under the `Server` key, locate a DWORD value named `Enabled` or create if it doesn't exist and set its value to "0".

External References

- How to disable TLS v1.0
- OWASP - Insecure Configuration Management
- OWASP - Insufficient Transport Layer Protection
- How to disable PCT 1.0, SSL 2.0, SSL 3.0, or TLS 1.0 in Internet Information Services
- IIS Crypto is a free tool that gives administrators the ability to enable or disable protocols, ciphers, hashes and key exchange algorithms on Windows Server 2003, 2008 and 2012
- Date Change for Migrating from SSL and Early TLS
- Browser Exploit Against SSL/TLS Attack (BEAST)

Classification

OWASP 2013-A6 PCI V3.1-6.5.4 PCI V3.2-6.5.4 CWE-327 CAPEC-217 WASC-4

5.1. Confirmed

Request

Response

6. [Possible] Cross-site Request Forgery

1 TOTAL
LOW

Netsparker identified a possible Cross-Site Request Forgery.

CSRF is a very common vulnerability. It's an attack which forces a user to execute unwanted actions on a web application in which the user is currently authenticated.

Impact

Depending on the application, an attacker can mount any of the actions that can be done by the user such as adding a user, modifying content, deleting data. All the functionality that's available to the victim can be used by the attacker. Only exception to this rule is a page that requires extra information that only the legitimate user can know (such as user's password).

Remedy

- Send additional information in each HTTP request that can be used to determine whether the request came from an authorized source. This "validation token" should be hard to guess for attacker who does not already have access to the user's account. If a request is missing a validation token or the token does not match the expected value, the server should reject the request.
- If you are posting form in ajax request, custom HTTP headers can be used to prevent CSRF because the browser prevents sites from sending custom HTTP headers to another site but allows sites to send custom HTTP headers to themselves using XMLHttpRequest.
    - For native XMLHttpRequest (XHR) object in JavaScript;

```
xhr = new XMLHttpRequest();
xhr.setRequestHeader('custom-header', 'value');
```

For JQuery, if you want to add a custom header (or set of headers) to

a. **individual request**

```
$.ajax({
    url: 'foo/bar',
    headers: { 'x-my-custom-header': 'some value' }
});
```

b. **every request**

```
$.ajaxSetup({
    headers: { 'x-my-custom-header': 'some value' }
});
OR
$.ajaxSetup({
    beforeSend: function(xhr) {
        xhr.setRequestHeader('x-my-custom-header', 'some value');
    }
});
```

External References

- OWASP Cross-Site Request Forgery (CSRF)

Remedy References

- OWASP Cross-Site Request Forgery (CSRF) Prevention Cheat Sheet

Classification

OWASP 2013-A8 PCI V3.1-6.5.9 PCI V3.2-6.5.9 CWE-352 CAPEC-62 WASC-9 HIPAA-164.306(A)

6.1.

Certainty

Request

Response

7. Mixed Content over HTTPS

1 TOTAL
LOW
CONFIRMED
1

Netsparker detected a mixed content loaded over HTTP within an HTTPS page.

Impact

If the HTTPS page includes content retrieved through regular, cleartext HTTP, then the connection is only partially encrypted. The unencrypted content is accessible to sniffers.

A man-in-the-middle attacker can intercept the request for the HTTP content and also rewrite the response to include malicious JavaScript code. Malicious active content can steal the user's credentials, acquire sensitive data about the user, or attempt to install malware on the user's system (by leveraging vulnerabilities in the browser or its plugins, for example), and therefore the connection is not safeguarded anymore.

Remedy

There are two technologies to defense against the mixed content issues:

1. HTTP Strict Transport Security (HSTS) is a mechanism that enforces secure resource retrieval, even in the face of user mistakes (attempting to access your web site on port 80) and implementation errors (your developers place an insecure link into a secure page)
2. Content Security Policy (CSP) can be used to block insecure resource retrieval from third-party web sites

Last but not least, you can use "protocol relative URLs" to have the user's browser automatically choose HTTP or HTTPS as appropriate, depending on which protocol the user is connected with. For example;

a protocol relative URL to load an image would look like <img src="//example.com/image.png">. The browser will automatically add either "http:" or "https:" to the start of the URL, whichever is appropriate.

External References

- Mixed Content

Remedy References

- Wikipedia - HTTP Strict Transport Security
- Wikipedia - Content Security Policy

Classification

7.1. Confirmed

Resources Loaded from Insecure Origin (HTTP)

```
http://www.w3schools.com/lib/w3.css
```

Request
Response

8. Forbidden Resource

1 TOTAL
INFORMATION
CONFIRMED
1

Netsparker identified a forbidden resource.

Access to this resource has been denied by the web server. This is generally not a security issue, and is reported here for informational purposes.

Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

Classification

OWASP-PC-C8

8.1. Confirmed

Request

Response

9. File Upload Functionality Detected

1 TOTAL
INFORMATION
CONFIRMED
1

Netsparker detected file upload functionality, which allows users to upload files to the web server.

Upload forms are generally dangerous, unless they are coded with a great deal of care. If there is any other vulnerability identified regarding this resource, Netsparker will report it as a separate issue.

Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

Classification

OWASP-PC-C4

9.1. Confirmed

Request

Response

10. ASP.NET Identified

1 TOTAL
INFORMATION

Netsparker identified that the target website is using ASP.NET as its web application framework.

This issue is reported as extra information only.

Impact

This issue is reported as additional information only. There is no direct impact arising from this issue.

Classification

OWASP-PC-C7

CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N/E:H/RL:O/RC:C
Base: 5.3 (Medium)
Temporal: 5.1 (Medium)
Environmental: 5.1 (Medium)

10.1.

Certainty

Request

Response

11. Email Address Disclosure

1 TOTAL
INFORMATION

Netsparker identified an email address disclosure.

Impact
Email addresses discovered within the application can be used by both spam email engines and also brute-force tools. Furthermore, valid email addresses may lead to social engineering attacks.

Remedy
Use generic email addresses such as contact@ or info@ for general communications and remove user/people-specific email addresses from the website; should this be required, use submission forms for this purpose.

External References
- Wikipedia - Email Spam

Classification
CWE-200 CAPEC-118 WASC-13 OWASP-PC-C7

CVSS 3.0
CVSS Vector String: CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Base: 5.3 (Medium)
Temporal: 5.3 (Medium)
Environmental: 5.3 (Medium)

11.1.

Certainty

Request

Response

12. Version Disclosure (IIS)

1 TOTAL
INFORMATION

Netsparker identified a version disclosure (IIS) in target web server's HTTP response.

This information can help an attacker gain a greater understanding of the systems in use and potentially develop further attacks targeted at the specific version of IIS.

Impact

An attacker might use the disclosed information to harvest specific security vulnerabilities for the version identified.

Remedy

Configure your web server to prevent information leakage from the `SERVER` header of its HTTP response.

Remedy References

- URLScan RemoveServerHeader Directive

Classification

CWE-205 CAPEC-170 WASC-45 HIPAA-164.306(A), 164.308(A) OWASP-PC-C7

12.1.

Certainty

Request

Response

13. HTTP Strict Transport Security (HSTS) Policy Not Enabled

1 TOTAL
INFORMATION

Netsparker identified that HTTP Strict Transport Security (HSTS) policy is not enabled.

The target website is being served from not only HTTP but also HTTPS and it lacks of HSTS policy implementation.

HTTP Strict Transport Security (HSTS) is a web security policy mechanism whereby a web server declares that complying user agents (such as a web browser) are to interact with it using only secure HTTP (HTTPS) connections. The HSTS Policy is communicated by the server to the user agent via a HTTP response header field named "Strict-Transport-Security". HSTS Policy specifies a period of time during which the user agent shall access the server in only secure fashion.

When a web application issues HSTS Policy to user agents, conformant user agents behave as follows:

- Automatically turn any insecure links referencing the web application into secure links. (For instance, http://example.com/some/page/ will be modified to https://example.com/some/page/ before accessing the server.)
- If the security of the connection cannot be ensured (e.g. the server's TLS certificate is self-signed), show an error message and do not allow the user to access the web application.

Remedy

Configure your webserver to redirect HTTP requests to HTTPS.

For Apache, you should have modification in the httpd.conf.

```
# load module
LoadModule headers_module modules/mod_headers.so

# redirect all HTTP to HTTPS (optional)
<VirtualHost *:80>
       ServerAlias *
       RewriteEngine On
       RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [redirect=301]
</VirtualHost>

# HTTPS-Host-Configuration
<VirtualHost *:443>
      # Use HTTP Strict Transport Security to force client to use secure connections only
      Header always set Strict-Transport-Security "max-age=31536000; includeSubDomains"

      # Further Configuration goes here
      [...]
</VirtualHost>
```

External References

- Wikipedia - HTTP Strict Transport Security
- Configure HSTS (HTTP Strict Transport Security) for Apache/Nginx

Classification

OWASP-PC-C8

13.1.

Certainty

Request

Response

14. OPTIONS Method Enabled

1 TOTAL
INFORMATION
CONFIRMED
1

Netsparker detected that `OPTIONS` method is allowed. This issue is reported as extra information.

Impact
Information disclosed from this page can be used to gain additional information about the target system.

Remedy
Disable `OPTIONS` method in all production systems.

External References

- Testing for HTTP Methods and XST (OWASP-CM-008)
- HTTP/1.1: Method Definitions

Classification
OWASP 2013-A5 CWE-16 CAPEC-107 WASC-14

14.1. Confirmed

Request
Response

15. Autocomplete Enabled (Password Field)

1 TOTAL
INFORMATION
CONFIRMED
1

Netsparker detected that autocomplete is enabled in one or more of the password fields.

Impact

If user chooses to save, data entered in these fields will be cached by the browser. An attacker who can access the victim's browser could steal this information. This is especially important if the application is commonly used in shared computers, such as cyber cafes or airport terminals.

Actions to Take

1. Add the attribute `autocomplete="off"` to the form tag or to individual "input" fields. However, since early 2014, major browsers don't respect this instruction, due to their integrated password management mechanism, and offer to users to store password internally.
2. Re-scan the application after addressing the identified issues to ensure all of the fixes have been applied properly.

Required Skills for Successful Exploitation

First and foremost, attacker needs either physical access or user-level code execution rights for successful exploitation. Dumping all data from a browser can be fairly easy, and a number of automated tools exist to undertake this. Where the attacker cannot dump the data, he/she could still browse the recently visited websites and activate the autocomplete feature to see previously entered values.

External References

- Using Autocomplete in HTML Forms
- How to Turn Off Form Autocompletion

Classification

OWASP 2013-A5 CWE-16 WASC-15

CVSS 3.0

CVSS Vector String: CVSS:3.0/AV:P/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N
Base: 4.6 (Medium)
Temporal: 4.6 (Medium)
Environmental: 4.6 (Medium)

15.1. Confirmed

Request
Response

16. Out-of-date Version (jQuery)

1 TOTAL
INFORMATION

Netsparker identified the target web site is using jQuery and detected that it is out of date.

Impact
Since this is an old version of the software, it may be vulnerable to attacks.

Remedy

Please upgrade your installation of jQuery to the latest stable version.

Remedy References

- Downloading jQuery

Classification
OWASP 2013-A9 PCI V3.1-6.2 PCI V3.2-6.2 CAPEC-310 OWASP-PC-C1

16.1.

Certainty

Request

Response

17. Missing X-XSS Protection Header

1 TOTAL
INFORMATION

Netsparker detected a missing `X-XSS-Protection` header which means that this website could be at risk of a Cross-site Scripting (XSS) attacks.

Impact
This issue is reported as additional information only. There is no direct impact arising from this issue.

Remedy
Add the X-XSS-Protection header with a value of "1; mode= block".

- `X-XSS-Protection: 1; mode=block`

External References

- MSDN - Internet Explorer 8 Security Features
- Internet Explorer 8 XSS Filter

Classification
HIPAA-164.308(A) OWASP-PC-C9

17.1.

Certainty

Request

Response

18. SameSite Cookie Not Implemented

1 TOTAL
INFORMATION
CONFIRMED
1

Cookies are typically sent to third parties in cross origin requests. This can be abused to do CSRF attacks. Recently a new cookie attribute named *SameSite* was proposed to disable third-party usage for some cookies, to prevent CSRF attacks.

Same-site cookies allow servers to mitigate the risk of CSRF and information leakage attacks by asserting that a particular cookie should only be sent with requests initiated from the same registrable domain.

Remedy

The server can set a same-site cookie by adding the SameSite=... attribute to the Set-Cookie header:

```
Set-Cookie: key=value; SameSite=strict
```

There are two possible values for the same-site attribute:

- Lax
- Strict

In the strict mode, the cookie is not sent with any cross-site usage even if the user follows a link to another website. Lax cookies are only sent with a top-level get request.

External References

- Using the Same-Site Cookies Attribute to Prevent CSRF Attacks
- Same-site Cookies
- Preventing CSRF with the same-site cookie attribute

Classification

OWASP-PC-C9

18.1. Confirmed

Request

Response

19. Subresource Integrity (SRI) Not Implemented

1 TOTAL
INFORMATION

Subresource Integrity (SRI) provides a mechanism to check integrity of the resource hosted by third parties like Content Delivery Networks (CDNs) and verifies that the fetched resource has been delivered without unexpected manipulation.

SRI does this using hash comparison mechanism. In this way, hash value declared in HTML elements (for now only script and link elements are supported) will be compared with the hash value of the resource hosted by third party.

Use of SRI is recommended as a best-practice, whenever libraries are loaded from a third-party source.

Remedy

Using Subresource Integrity is simply to add *integrity* attribute to the *script* tag along with a base64 encoded cryptographic hash value.

```
<script src="https://code.jquery.com/jquery-2.1.4.min.js" integrity="sha384-R4/ztc4ZlRqWjqIuvf6RX5yb/v90qNGx6fS48N0tRxiGkqveZETq72KgDVJCp2TC" crossorigin="anonymous"></script>
```

The hash algorithm must be one of **sha256**, **sha384** or **sha512**, followed by a '-' character.

External References

- Subresource Integrity
- Do not let your CDN betray you: Use Subresource Integrity
- Web Application Security with Subresource Integrity
- SRI Hash Generator

Classification

19.1.

Certainty

Request

Response

20. Content Security Policy (CSP) Not Implemented

1 TOTAL
INFORMATION

CSP is an added layer of security that helps to mitigate mainly Cross-site Scripting attacks.

CSP can be enabled instructing the browser with a Content-Security-Policy directive in a response header;

```
Content-Security-Policy: script-src 'self';
```

or in a meta tag;

```
<meta http-equiv="Content-Security-Policy" content="script-src 'self';">
```

In the above example, you can restrict script loading only to the same domain. It will also restrict inline script executions both in the element attributes and the event handlers. There are various directives which you can use by declaring CSP:

- **script-src:** Restricts the script loading resources to the ones you declared. By default, it disables inline script executions unless you permit to the evaluation functions and inline scripts by the unsafe-eval and unsafe-inline keywords.
- **base-uri:** Base element is used to resolve relative URL to absolute one. By using this CSP directive, you can define all possible URLs which could be assigned to base-href attribute of the document.
- **frame-ancestors**: It is very similar to X-Frame-Options HTTP header. It defines the URLs by which the page can be loaded in an iframe.
- **frame-src / child-src**: frame-src is the deprecated version of child-src. Both define the sources that can be loaded by iframe in the page. (Please note that frame-src was brought back in CSP 3)
- **object-src** : Defines the resources that can be loaded by embedding such as Flash files, Java Applets.
- **img-src**: As its name implies, it defines the resources where the images can be loaded from.
- **connect-src**: Defines the whitelisted targets for XMLHttpRequest and WebSocket objects.
- **default-src**: It is a fallback for the directives that mostly ends with -src suffix. When the directives below are not defined, the value set to default-src will be used instead:
  - child-src
  - connect-src
  - font-src
  - img-src
  - manifest-src
  - media-src
  - object-src
  - script-src
  - style-src

When setting the CSP directives, you can also use some CSP keywords:

- **none**: Denies loading resources from anywhere.
- **self** : Points to the document's URL (domain + port).
- **unsafe-inline**: Permits running inline scripts.
- **unsafe-eval**: Permits execution of evaluation functions such as eval().

In addition to CSP keywords, you can also use wildcard or only a scheme when defining whitelist URLs for the points. Wildcard can be used for subdomain and port portions of the URLs:

```
Content-Security-Policy: script-src https://*.example.com;

Content-Security-Policy: script-src https://example.com:*;

Content-Security-Policy: script-src https;
```

It is also possible to set a CSP in Report-Only mode instead of forcing it immediately in the migration period. Thus you can see the violations of the CSP policy in the current state of your web site while migrating to CSP:

```
Content-Security-Policy-Report-Only: script-src 'self'; report-uri: https://example.com;
```

Impact

There is no direct impact of not implementing CSP on your website. However, if your website is vulnerable to a Cross-site Scripting attack CSP can prevent successful exploitation of that vulnerability. By not implementing CSP you'll be missing out this extra layer of security.

Actions to Take

- Enable CSP on your website by sending the `Content-Security-Policy` in HTTP response headers that instruct the browser to apply the policies you specified.
- Apply the whitelist and policies as strict as possible.
- Rescan your application to see if Netsparker identifies any weaknesses in your policies.

Remedy

Enable CSP on your website by sending the `Content-Security-Policy` in HTTP response headers that instruct the browser to apply the policies you specified.

External References

- An Introduction to Content Security Policy
- Content Security Policy (CSP)

Classification

OWASP-PC-C9

20.1.

Certainty

Request

Response