Security Headers & How to Enable/Insert Security Headers in APV

What are Security headers?

They are directives to increase the protection and create more defense against vulnerabilities using browsers.

How Security Headers Can Prevent Vulnerabilities?

Inserting a security header can prevent a variety of hacking attempts. Security headers can address a number of cyber threats. Also known as security-related HTTP response headers, they modify the behavior of web browsers to avoid security vulnerabilities.

Security Headers & How to Enable/Insert Security Headers?

1. HTTP Strict Transport Security (HSTS)

Description : HTTP Strict Transport Security (also named HSTS) is a web security policy mechanism which helps to protect websites against protocol downgrade attacks and cookie hijacking. It allows web servers to declare that web browsers (or other complying user agents) should only interact with it using secure HTTPS connections, and never via the insecure HTTP protocol. HSTS is an IETF standards track protocol and is specified in RFC 6797. A server implements an HSTS policy by supplying a header (`Strict-Transport-Security`) over an HTTPS connection (HSTS headers over HTTP are ignored).

Values

Value	Description
`max-age=SECONDS`	The time, in seconds, that the browser should remember that this site is only to be accessed using HTTPS.
`includeSubDomains`	If this optional parameter is specified, this rule applies to all of the site's subdomains as well.

Example

```
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
```

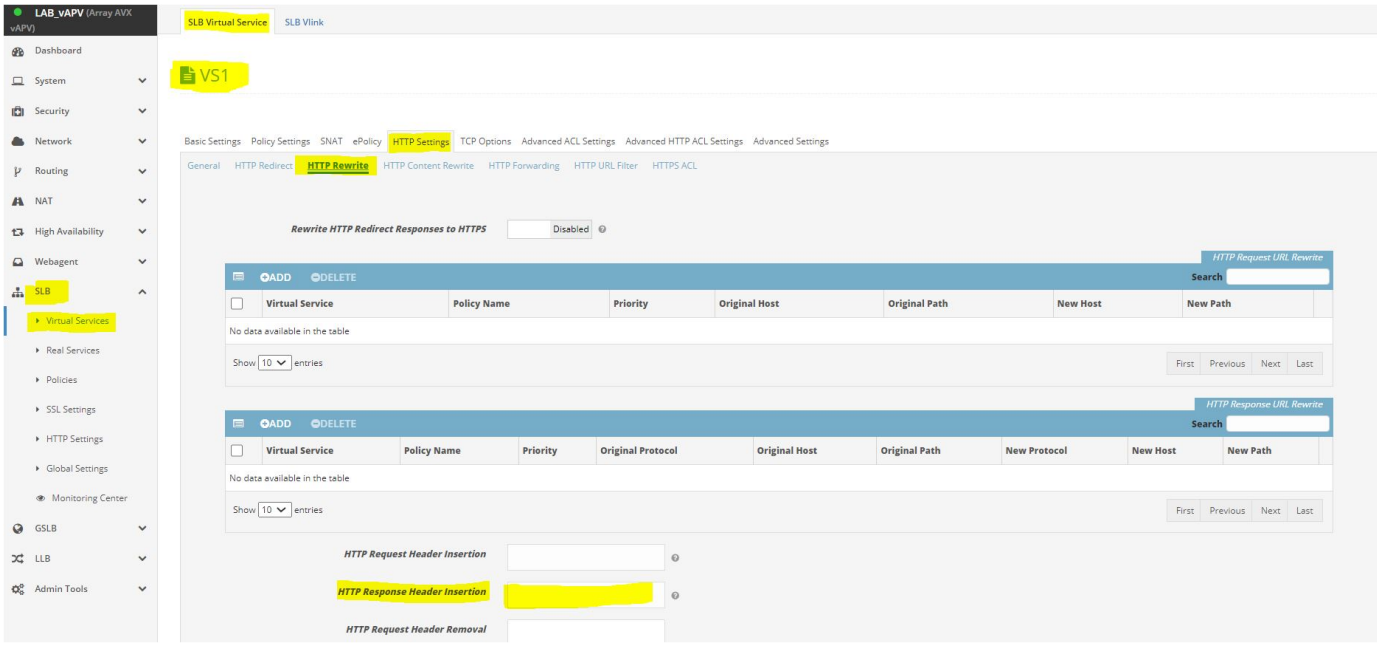
CLI commands to insert HSTS header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “Strict-Transport-Security: max-age=31536000; includeSubDomains“

Webui steps to insert HSTS header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Strict-Transport-Security: max-age=31536000; includeSubDomains and save the configuration)



2. X-Frame-Options

Description : The `X-Frame-Options` response header (also named XFO) improves the protection of web applications against clickjacking. It instructs the browser whether the content can be displayed within frames.
The CSP (Content Security Policy) frame-ancestors directive obsoletes the X-Frame-Options header. If a resource has both policies, the CSP frame-ancestors policy will be enforced and the X-Frame-Options policy will be ignored

Values

Value	Description
`deny`	No rendering within a frame.
`sameorigin`	No rendering if origin mismatch.
`allow-from: DOMAIN`	Allows rendering if framed by frame loaded from DOMAIN.

Example

```
X-Frame-Options: deny
```

CLI commands to insert X-Frame-Options header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “X-Frame-Options: deny“

Webui steps to insert X-Frame-Options header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header X-Frame-Options: deny and save the configuration)

3. X-Content-Type-Options

Description : Setting this header will prevent the browser from interpreting files as a different MIME type to what is specified in the `Content-Type` HTTP header (e.g. treating `text/plain` as `text/css`).

Values

Value	Description
`nosniff`	Will prevent the browser from MIME-sniffing a response away from the declared content-type.

Example

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

CLI commands to insert X-Content-Type-Options header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “X-Content-Type-Options: nosniff“

Webui steps to insert X-Content-Type-Options header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header X-Content-Type-Options: nosniff and save the configuration)

4. Content-Security-Policy

Description : A Content Security Policy (also named CSP) requires careful tuning and precise definition of the policy. If enabled, CSP has significant impact on the way browsers render pages (e.g., inline JavaScript is disabled by default and must be explicitly allowed in the policy). CSP prevents a wide range of attacks, including cross-site scripting and other cross-site injections.

Values

Value	Description
`base-uri`	Define the base URI for relative URIs.
`default-src`	Define loading policy for all resources type in case a resource type’s dedicated directive is not defined (fallback).
`script-src`	Define which scripts the protected resource can execute.
`object-src`	Define from where the protected resource can load plugins.
`style-src`	Define which styles (CSS) can be applied to the protected resource.
`img-src`	Define from where the protected resource can load images.
`media-src`	Define from where the protected resource can load video and audio.
`frame-src`	*(Deprecated and replaced by* `child-src`) Define from where the protected resource can embed frames.
`child-src`	Define from where the protected resource can embed frames.
`frame-ancestors`	Define from where the protected resource can be embedded in frames.
`font-src`	Define from where the protected resource can load fonts.
`connect-src`	Define which URIs the protected resource can load using script interfaces.

`manifest-src`	Define from where the protected resource can load manifests.
`form-action`	Define which URIs can be used as the action of HTML form elements.
`sandbox`	Specifies an HTML sandbox policy that the user agent applies to the protected resource.
`script-nonce`	Define script execution by requiring the presence of the specified nonce on script elements.
`plugin-types`	Define the set of plugins that can be invoked by the protected resource by limiting the types of resources that can be embedded.
`reflected-xss`	Instruct the user agent to activate or deactivate any heuristics used to filter or block reflected cross-site scripting attacks, equivalent to the effects of the non-standard `X-XSS-Protection` header.
`block-all-mixed-content`	Prevent the user agent from loading mixed content.
`upgrade-insecure-requests`	Instruct the user agent to download insecure HTTP resources using HTTPS.
`referrer`	*(Deprecated)* Define information the user agent can send in the `Referer` header.
`report-uri`	*(Deprecated and replaced by* `report-to`) Specifies a URI to which the user agent sends reports about policy violation.
`report-to`	Specifies a group (defined in the `Report-To` header) to which the user agent sends reports about policy violation.

Example

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

CLI commands to insert Content-Security-Policy header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “Content-Security-Policy: script-src ‘self’“

Webui steps to insert Content-Security-Policy header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Content-Security-Policy: script-src ‘self' and save the configuration)

5. X-Permitted-Cross-Domain-Policies

Description : A cross-domain policy file is an XML document that grants a web client, such as Adobe Flash Player or Adobe Acrobat (though not necessarily limited to these), permission to handle data across domains. When clients request content hosted on a particular source domain and that content makes requests directed towards a domain other than its own, the remote domain needs to host a cross-domain policy file that grants access to the source domain, allowing the client to continue the transaction. Normally a meta-policy is declared in the master policy file, but for those who can’t write to the root directory, they can also declare a meta-policy using the `X-Permitted-Cross-Domain-Policies` HTTP response header.

Values

Value	Description
`none`	No policy files are allowed anywhere on the target server, including this master policy file.

`master-only`	Only this master policy file is allowed.
`by-content-type`	[HTTP/HTTPS only] Only policy files served with Content-Type: text/x-cross-domain-policy are allowed.
`by-ftp-filename`	[FTP only] Only policy files whose file names are crossdomain.xml (i. e. URLs ending in /crossdomain.xml) are allowed.
`all`	All policy files on this target domain are allowed.

Example

```
X-Permitted-Cross-Domain-Policies: none
```

CLI commands to insert X-Permitted-Cross-Domain-Policies header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “X-Permitted-Cross-Domain-Policies: none“

Webui steps to insert X-Permitted-Cross-Domain-Policies header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header X-Permitted-Cross-Domain-Policies: none and save the configuration)

6. Referrer-Policy

Description : The `Referrer-Policy` HTTP header governs which referrer information, sent in the `Referer` header, should be included with requests made.

Values

Value	Description
`no-referrer`	The `Referer` header will be omitted entirely. No referrer information is sent along with requests.
`no-referrer-when-downgrade`	This is the user agent’s default behavior if no policy is specified. The origin is sent as referrer to a-priori as-much-secure destination (HTTPS HTTPS), but isn’t sent to a less secure destination (HTTPS HTTP).
`origin`	Only send the origin of the document as the referrer in all cases. (e.g. the document `https://example.com/page.html` will send the referrer `https://example.com/`.)
`origin-when-cross-origin`	Send a full URL when performing a same-origin request, but only send the origin of the document for other cases.
`same-origin`	A referrer will be sent for same-site origins, but cross-origin requests will contain no referrer information.
`strict-origin`	Only send the origin of the document as the referrer to a-priori as-much-secure destination (HTTPS HTTPS), but don’t send it to a less secure destination (HTTPS HTTP).
`strict-origin-when-cross-origin`	Send a full URL when performing a same-origin request, only send the origin of the document to a-priori as-much-secure destination (HTTPS HTTPS), and send no header to a less secure destination (HTTPS HTTP).
`unsafe-url`	Send a full URL (stripped from parameters) when performing a same-origin or cross-origin request.

Example

```
Referrer-Policy: no-referrer
```

CLI commands to insert Referrer-Policy header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader “VS1“ “Referrer-Policy: no-referrer“

Webui steps to insert Referrer-Policy header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Referrer-Policy: no-referrer and save the configuration)

7. Clear-Site-Data

Description : The Clear-Site-Data header clears browsing data (cookies, storage, cache) associated with the requesting website. It allows web developers to have more control over the data stored locally by a browser for their origins. This header is useful for example, during a logout process, in order to ensure that all stored content on the client side like cookies, storage and cache are removed.

Values

Value	Description
`"cache"`	Indicates that the server wishes to remove locally cached data for the origin of the response URL.
`"cookies"`	Indicates that the server wishes to remove all cookies for the origin of the response URL. HTTP authentication credentials are also cleared out. This affects the entire registered domain, including subdomains.
`"storage"`	Indicates that the server wishes to remove all DOM storage for the origin of the response URL.
`"executionContexts"`	Indicates that the server wishes to reload all browsing contexts for the origin of the response. Currently, this value is only supported by a small subset of browsers.
`"*"`	Indicates that the server wishes to clear all types of data for the origin of the response. If more data types are added in future versions of this header, they will also be covered by it.

Example

```
Clear-Site-Data: "cache","cookies","storage"
```

CLI commands to insert Clear-Site-Data header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "Clear-Site-Data: %qcache%q,%qcookies%q"

Webui steps to insert Clear-Site-Data header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Clear-Site-Data: %qcache%q,%qcookies%q and save the configuration)

8. Cross-Origin-Embedder-Policy

Description : This response header (also named COEP) prevents a document from loading any cross-origin resources that don't explicitly grant the document permission.

Values

Value	Description
`unsafe-none`	Allows the document to fetch cross-origin resources without giving explicit permission through the CORS protocol or Cross-Origin-Resource-Policy header (it is the default value).
`require-corp`	A document can only load resources from the same origin, or resources explicitly marked as loadable from another origin.

Example

```
Cross-Origin-Embedder-Policy: require-corp
```

CLI commands to insert Cross-Origin-Embedder-Policy header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "Cross-Origin-Embedder-Policy: require-corp"

Webui steps to insert Cross-Origin-Embedder-Policy header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Cross-Origin-Embedder-Policy: require-corp and save the configuration)

9. Cross-Origin-Opener-Policy

Description : This response header (also named COOP) allows you to ensure a top-level document does not share a browsing context group with cross-origin documents. COOP will process-isolate your document and potential attackers can't access to your global object if they were opening it in a popup, preventing a set of cross-origin attacks dubbed.

Values

Value	Description
`unsafe-none`	Allows the document to be added to its opener's browsing context group unless the opener itself has a COOP of `same-origin` or `same-origin-allow-popups` (it is the default value).
`same-origin-allow-popups`	Retains references to newly opened windows or tabs which either don't set COOP or which opt out of isolation by setting a COOP of `unsafe-none`.
`same-origin`	Isolates the browsing context exclusively to same-origin documents. Cross-origin documents are not loaded in the same browsing context.

Example

```
Cross-Origin-Opener-Policy: same-origin
```

CLI commands to insert Cross-Origin-Opener-Policy header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "Cross-Origin-Opener-Policy: same-origin"

Webui steps to insert Cross-Origin-Opener-Policy header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Cross-Origin-Opener-Policy: same-origin and save the configuration)

10. Cross-Origin-Resource-Policy

Description : This response header (also named CORP) allows to define a policy that lets web sites and applications opt in to protection against certain requests from other origins (such as those issued with elements like `<script>` and `<img>`), to mitigate speculative side-channel attacks, like Spectre, as well as Cross-Site Script Inclusion (XSSI) attacks.

Values

Value	Description
`same-site`	Only requests from the same Site can read the resource.
`same-origin`	Only requests from the same Origin (i.e. scheme + host + port) can read the resource.
`cross-origin`	Requests from any Origin (both `same-site` and `cross-site`) can read the resource. Browsers are using this policy when an CORP header is not specified.

Example

```
Cross-Origin-Resource-Policy: same-origin
```

CLI commands to insert Cross-Origin-Resource-Policy header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "Cross-Origin-Resource-Policy: same-origin"

Webui steps to insert Cross-Origin-Resource-Policy header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Cross-Origin-Resource-Policy: same-origin and save the configuration)

11. Cache-Control

Description : This header holds directives (instructions) for caching in both **requests** and **responses**. If a given directive is in a request, it does not mean this directive is in the response. Specify the capability of a resource to be cached is important to prevent exposure of information via the cache.
The headers named Expires and Pragma can be used in addition to the Cache-Control header. Pragma header can be used for backwards compatibility with the HTTP/1.0 caches. However, *Cache-Control* is the recommended way to define the caching policy.

Values applicable for HTTP responses

Value	Description
`must-revalidate`	Indicates that once a resource becomes stale, caches do not use their stale copy without successful validation on the origin server.

`no-cache`	The response may be stored by any cache, even if the response is normally non-cacheable. However, the stored response **MUST always** go through validation with the origin server first before using it.
`no-store`	The response may not be stored in any cache.
`no-transform`	An intermediate cache or proxy cannot edit the response body, `Content-Encoding`, `Content-Range`, or `Content-Type`.
`public`	The response may be stored by any cache, even if the response is normally non-cacheable.
`private`	The response may be stored only by a browser's cache, even if the response is normally non-cacheable.
`proxy-revalidate`	Like `must-revalidate`, but only for shared caches (e.g., proxies). Ignored by private caches.
`max-age=<seconds>`	The maximum amount of time a resource is considered fresh. Unlike Expires, this directive is relative to the time of the request.
`s-maxage=<seconds>`	Overrides `max-age` or the Expires header, but only for shared caches (e.g., proxies). Ignored by private caches.

Example

No caching allowed, clear any previously cached resources and include support for HTTP/1.0 caches:

```
Cache-Control: no-store, max-age=0
```

Caching allowed with a cache duration of one week:

```
Cache-Control: public, max-age=604800
```

CLI commands to insert Cache-Control header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "Cache-Control: public, max-age=604800"

Webui steps to insert Cache-Control header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header Cache-Control: public, max-age=604800 and save the configuration)

12. X-XSS-Protection

Description : he X-XSS-Protection header has been deprecated by modern browsers and its use can introduce additional security issues on the client side. As such, it is recommended to set the header as `X-XSS-Protection: 0` in order to disable the XSS Auditor, and not allow it to take the default behavior of the browser handling the response.
This header enables the cross-site scripting (XSS) filter in your browser.

Values

Value	Description
`0`	Filter disabled.

`1`	Filter enabled. If a cross-site scripting attack is detected, in order to stop the attack, the browser will sanitize the page.
`1; mode=block`	Filter enabled. Rather than sanitize the page, when a XSS attack is detected, the browser will prevent rendering of the page.
`1; report=http://[YOURDOMAIN]/your_report_URI`	Filter enabled. The browser will sanitize the page and report the violation. This is a Chromium function utilizing CSP violation reports to send details to a URI of your choice.

Example

```
X-XSS-Protection: 0
```

CLI commands to insert X-XSS-Protection header:

http rewrite response insertheader <Virtual_Service> <Header_String>

Example:
http rewrite response insertheader "VS1" "X-XSS-Protection: 0"

Webui steps to insert X-XSS-Protection header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header X-XSS-Protection: 0 and save the configuration)

NOTE:

1. **CLI command and Webui steps to add/insert header is same for any of the header mentioned above. Only "Header_Strings" differs.**
2. **If we want to add/insert multiple header then, we need to use %n sign to separate headers as shown in below example.**

Example:
Adding 3 headers together (Strict-Transport-Security , X-Frame-Options , X-XSS-Protection)

CLI command to insert multiple headers:

http rewrite response insertheader "VS1" "X-XSS-Protection: 0%nX-Frame-Options: deny%nStrict-Transport-Security: max-age=31536000"

Webui steps to insert X-XSS-Protection header:

Login to Webui SLB Select Virtual service HTTP Settings HTTP Rewrite ***HTTP Response Header Insertion*** (Enter the header X-XSS-Protection: 0%nX-Frame-Options: deny%nStrict-Transport-Security: max-age=31536000 and save the configuration)

==========**END OF DOCUMENT**==========