**HTTP Response Headers**

**SEVERITY:** **MODERATE**

**VULN. CODE:** RES-H

**AUTH. REQUIRED:**

**VULNERABILITY DESCRIPTION**

Modern web applications require modern defense and reinforcement systems. With the progress of the sophistication of web based applications, user browsers have also become a major component in mitigating client-side prosecution attacks. The HTTP protocol (Hyper Text Transfer Protocol) is a protocol without nature of the state control. Currently, with the development of web technologies, the HTTP protocol, which has become the backbone of the entire World Wide Web, has evolved to the point of being able to manage and maintain the state of the sessions of multiple contemporary users who use the same web application. This is made possible mainly through the implementation of session controls via cookies. The multitude of attacks that can be prosecuted via HTTP can lead to exploit the exchange of important information between the user's browser and the web server on every single request and response. Therefore, the setting and strengthening of HTTP headers, on the web server side, is a good basis for preventing and mitigating client-side attacks (i.e. aimed at end users' browsers).

During the analysis, it was detected that the web application responds to client HTTP requests without implementing useful controls within the protocol header. In particular, within the HTTP RESPONSE coming from the analyzed web application, the following HTTP headers are not configured:

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

The Strict-Transport-Security HTTP header (HSTS) strengthens secure SSL / TLS connections. This setting, on the web server side, reduces the impact that any bugs present in the applications make possible the loss of session information, through cookies and external links, eliminating any possibility of passing in clear on the network of such information. The transport in cleartext on the network can facilitate Man-in-the-middle attacks, giving rise to attacks aimed at the theft of the session and the interception of the contents. The HSTS header is also set to inhibit the ability of the user-browser side to ignore the warnings arising during the negotiation phase of the SSL / TLS certificates, thus rejecting the connection in case of errors or inconsistencies on the certificates. The parameter max-age is the time in seconds for which the enforcement policy on the user's browser will be valid.

X-XSS-Protection: 1; mode=block

This HTTP header, set on the web server side, forces all Cross-Site Scripting (XSS) filters to be enabled on modern web browsers, even if the user had on purpose disabled these controls.

X-Content-Type-Options: nosniff

This header is particularly important to instruct modern browsers in order to prevent the interception of MIME contents. This web server-side setting reduces exposure to attacks carried by requests for downloading and uploading content that dynamic HTML pages and executables can pursue without the knowledge of the end user.

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

The Content-Security-Policy header extends the Same Origin Policy rule, introducing a Whitelist concept allowing to specify precise rules on the domains and Cross-scripting sites that are accepted. With this system you can individually specify accepted urls for iframes, for image css scripts and much more. So the header Content-Security-Policy must be associated with a parameter through which to indicate the URL of the CDN (Content Delivery Network) authorized to make available its libraries. Through the proposed header the only accepted scripts will be those available through the current (self) domain. Furthermore, the directive can be specified for:

* script-src: JavaScript code;
* connect-src: XMLHttpRequest, WebSockets e EventSource;
* font-src: fonts;
* frame-src: frame;
* img-src: images;
* media-src: audio/video
* object-src: multimedia plugins like Adobe Flash;
* style-src: CSS sheets.

|  |
| --- |
| **RESPONSE https://11.22.33.44/**  HTTP/1.1 200 OK  Cache-Control: no-cache  Cache-Control: no-store  Content-Language: it  Date: Fri, 17 May 2019 11:56:53 GMT  Expires: Thu, 01 Jan 1970 00:00:00 GMT  Pragma: no-cache  Server: nginx/1.4.6 (Ubuntu)  Connection: Close  Content-Length: 20405 |

**Box 1 – Missing HTTP Response Headers**

**SUGGESTED SOLUTIONS**

We recommend, where possible, to implement these settings on the web server side, in order to prevent a large class of attacks directed at the user client, that is targeted attacks on end users who use the web application at a given moment. It is advisable to remove the "X-Powered-By:" information header from which information about the software components in question can be easily obtained. Alternatively, it is possible to specify a fictitious value that has no specific reference and that can not external threat to installed software components.

After the modification and restart of the WEB server, all the headers with the set values ​​will be contained within all the responses (RESPONSE HTTP) by the server. These headers will be interpreted by the end user's browser that will automatically prevent the user's browser from operations that could be automatically recalled without the user's knowledge from pages properly constructed for fraud and / or other malicious uses.

|  |
| --- |
| X-Content-Type-Options: nosniff  Content-Security-Policy: default-src 'self'; script-src 'self'  X-Frame-Options: DENY  X-XSS-Protection: 1; mode=block  Strict-Transport-Security: max-age=31536000; includeSubDomains |

**REFERENCES**

More information regarding the vulnerability and its possible solutions can be found on the following addresses:

**REF 1 – www.owasp.org/index.php/List\_of\_useful\_HTTP\_headers**

**REF 2 – securityheaders.com/**

**REF 3 – blogs.msdn.com/b/ie/archive/2008/09/02/ie8-security-part-vi-beta-2-update.aspx**

**REF 4 – blogs.msdn.com/b/ie/archive/2008/07/02/ie8-security-part-iv-the-xss-filter.aspx**

**REF 5 – tools.ietf.org/html/rfc6797**

**REF 6 – hackertarget.com/http-header-security-analysis/**

**REF 7 – www.html5rocks.com/en/tutorials/security/content-security-policy/**