International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
***Article Received:** 25 May 2023 **Revised:** 12 June 2023 **Accepted:** 30 July 2023*

Strengthening E-Learning Security: A Study on the Implementation and Efficacy of HTTP Security Headers in e-learning platforms

Saroj Junghare,
Research Scholar, Rani Durgavati Vishwavidyalaya, Jabalpur, Madhya Pradesh.

Prof. Mridula Dube,

Director, UICSA, Rani Durgavati Vishwavidyalaya, Jabalpur, Madhya Pradesh. Corresponding Author E-mail Id:

kanojiasaroj8384@gmail.com

Abstract-In today's digital environment, securing web applications is more important than ever. As web technologies become increasingly complex and interconnected, the risks and vulnerabilities that threaten sensitive information and user privacy also rise. Among the different strategies available to improve web security, the use of security headers is particularly significant, even though it is frequently overlooked. As web technologies become increasingly sophisticated and interconnected, the risks and vulnerabilities that threaten sensitive information and user privacy also escalate. This is particularly crucial for e-learning platforms, where safeguarding personal and academic data is vital for creating a secure and reliable environment for users.

Keywords: Security Headers, E-Learning, Information Security, Web Security.

1. Introduction

The Core Security Mechanisms which needs to be applied on any E-learning platforms are as follows: Authentication and Authorization Mechanisms (Strong Password Policies, Multi- Factor Authentication (MFA), Role-Based Access Control (RBAC), Single Sign-On (SSO)), Data Encryption Mechanisms (Data in Transit like Transport Layer Security (TLS), Data at Rest), Input Validation and Sanitization Mechanisms, Secure Coding Practices, Network Security and Web Application Firewalls (WAFs) by keeping in mind the following additional considerations like User Education and Awareness, Incident Response Plan, Compliance with Data Protection Regulations and Continuous Monitoring and Logging.

eLearning platforms require robust security measures on both the client and server sides. Security Headers are categorised into Client-side security headers and Server-side security header. Client-side security headers like Content Security Policy (CSP) restrict the resources a browser can load, preventing cross-site scripting (XSS) attacks. Strict Transport Security (HSTS) enforces HTTPS connections, safeguarding against man-in-the-middle attacks. Server-side headers like X-Frame-Options mitigate clickjacking, while HTTP Public Key Pinning (HPKP) protects against certificate spoofing. Together, these headers form a multi-layered defence against common web vulnerabilities, enhancing the overall security posture of eLearning platforms. As more students use these platforms, the valuable data transmitted and stored there becomes a target for attackers[1]. HTTP security headers are designed to prevent these general safety risks[7].

HTTP Security Headers refer to various pieces of information that a server sends to a user's browser. With the progress of the Internet, it will become more and more popular to provide a

wide range of services to end users on the Internet[2]. Their primary purpose is to enhance the security of web application, web server settings and API can greatly improve by offering instructions on how to manage the page and its resources. Moreover, web developers need to ensure these headers are properly configured to achieve optimal protection.

To improve security of any website, HTTP Security Headers send instructions to a user's browser regarding how to process the web page and its associated resources; this can establish a more secure connection between browser and the web server, helping to prevent vulnerabilities such as XSS and CSRF. Moreover, these headers can oversee cross-origin resource sharing, manage MIME types, enforce content security policies, and guard against click jacking attacks. It has been a recurrent research topic, aided by the fact that the nature of the World Wide Web makes data publicly accessible to any interested party and that the WWW itself is continuously growing and evolving[6].

HTTP security headers function at the runtime level, offering a significantly broader layer of protection. The web server is reinforced due to the appropriate configuration of security securities [3]. By limiting the actions allowed by the browser and server during the operation of the web application, these headers can prevent entire categories of attacks, which makes them highly effective. Implementing the appropriate headers correctly is vital for any application setup that follows best practices. However, it is essential first to select the headers that will have the most significant impact, followed by thorough implementation and testing throughout the application environment to ensure a balance between security and functionality. With the increase in the number of threats within web-based systems, a more integrated approach is required to ensure the enforcement of security policies from the server to the client[4].

1.1 How Security Headers work

Web applications are widely used, and the applications deployed on the web do not always satisfy all the security policies[5]. Security headers are instructions used by web applications to set up security measures within web. By following these instructions, browsers can help mitigate client-side vulnerabilities, including Cross-Site Scripting and Clickjacking. Moreover, headers can be used to ensure that browsers permit only legitimate TLS communication, enforce valid certificates, or even mandate the use of a specific server certificate. HTTP headers can serve as effective and cheap means by which websites can declare to a browser what security principles to enforce[8].

Security headers refer to HTTP response headers that your application can implement to bolster its security. Once set up, these headers can stop modern browsers from taking advantage of easily preventable vulnerabilities.

Security headers are directives that browsers are required to comply with, transmitted through the HTTP header response. They provide guidelines and restrictions to avoid unintended security breaches.

Sr. No.	Feature	Server-Side Security Headers	Client-Side Security Headers
1	Origin	Sent from the server to the client's browser	Implemented and enforced by the browser
2	Primary Goal	Protect the server and its resources from attacks	Protect the client's browser and user data from attacks
3	Examples of Headers	X-Frame-Options, HSTS, HPKP, X-Content-Type- Options	CSP, HSTS
4	Key Protections	Clickjacking, Man-in-the- Middle attacks, Certificate spoofing, MIME sniffing	XSS, Data exposure, Malicious script execution
5	Control Level	Server administrators have full control over the headers	Users can partially influence headers through browser extensions or configuration
6	Complexity	Often more complex to implement and manage	Generally simpler to implement and manage

Table 1: Security Header types and features

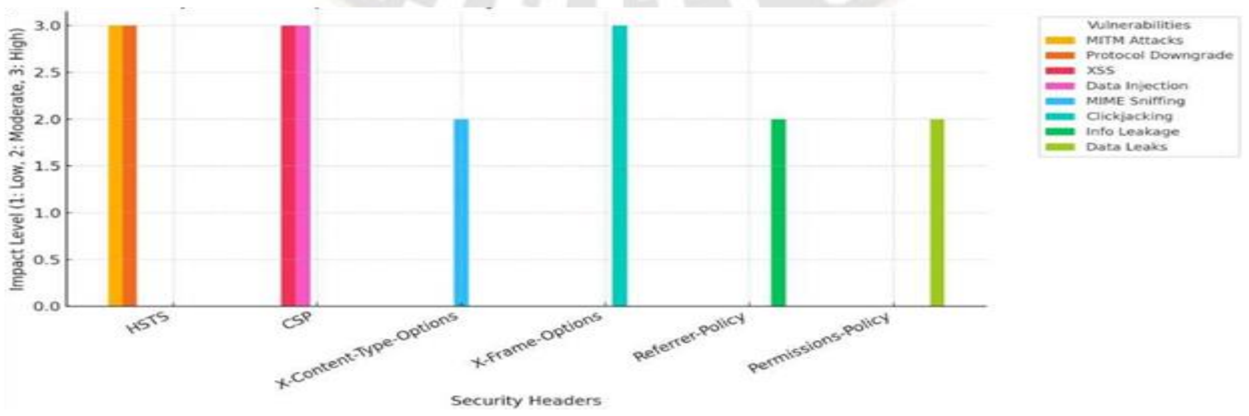
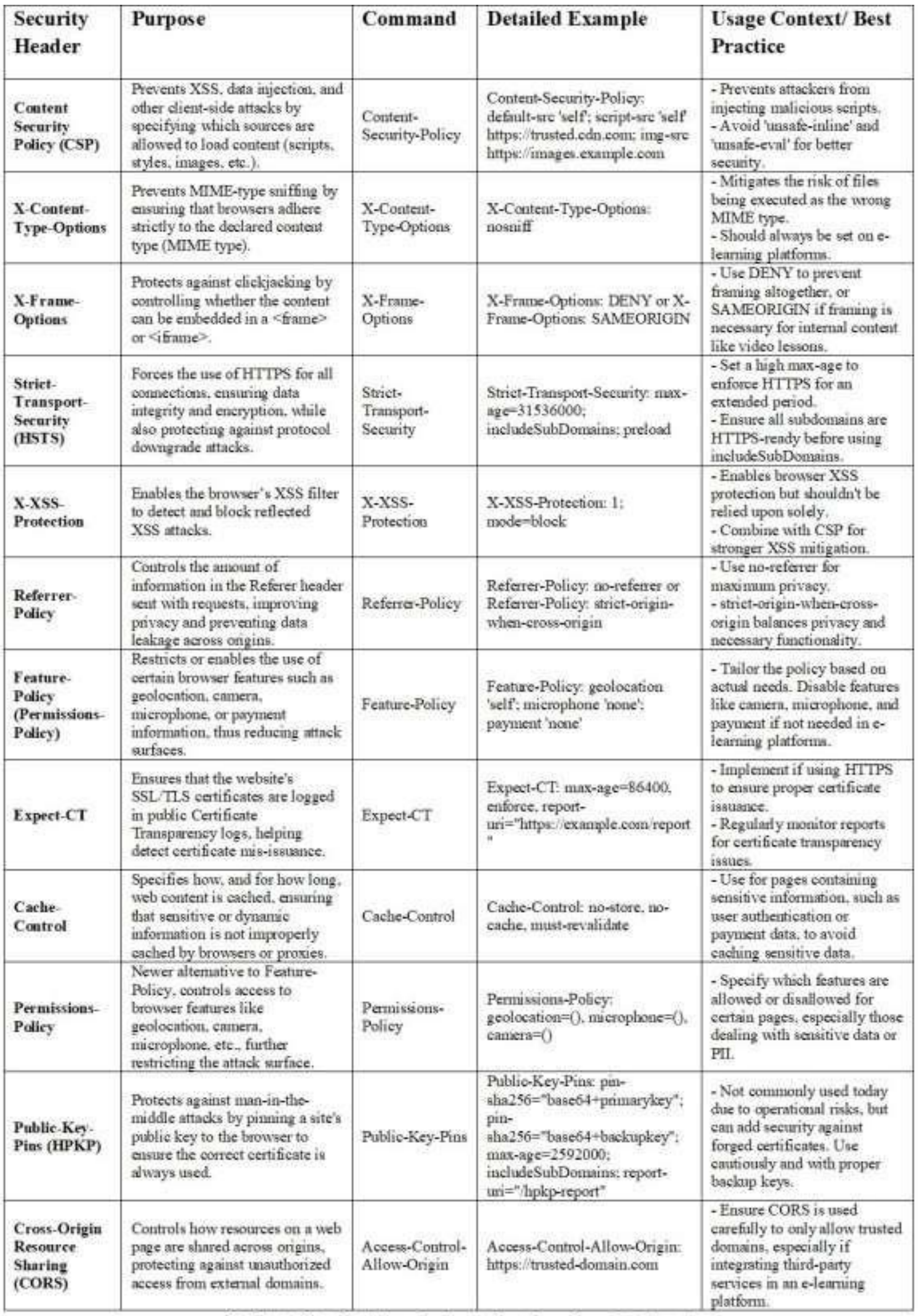


Chart 1: Comparative impact of Security Headers on Common Vulnerabilities



Security Header	Purpose	Command	Detailed Example	Usage Context/ Best Practice
Content Security Policy (CSP)	Prevents XSS, data injection, and other client-side attacks by specifying which sources are allowed to load content (scripts, styles, images, etc.).	Content-Security-Policy	Content-Security-Policy: default-src 'self'; script-src 'self' https://trusted.cdn.com; img-src https://images.example.com	- Prevents attackers from injecting malicious scripts. - Avoid 'unsafe-inline' and 'unsafe-eval' for better security.
X-Content-Type-Options	Prevents MIME-type sniffing by ensuring that browsers adhere strictly to the declared content type (MIME type).	X-Content-Type-Options	X-Content-Type-Options: nosniff	- Mitigates the risk of files being executed as the wrong MIME type. - Should always be set on e-learning platforms.
X-Frame-Options	Protects against clickjacking by controlling whether the content can be embedded in a <frame> or <iframe>.	X-Frame-Options	X-Frame-Options: DENY or X-Frame-Options: SAMEORIGIN	- Use DENY to prevent framing altogether, or SAMEORIGIN if framing is necessary for internal content like video lessons.
Strict-Transport-Security (HSTS)	Forces the use of HTTPS for all connections, ensuring data integrity and encryption, while also protecting against protocol downgrade attacks.	Strict-Transport-Security	Strict-Transport-Security: max-age=31536000; includeSubDomains; preload	- Set a high max-age to enforce HTTPS for an extended period. - Ensure all subdomains are HTTPS-ready before using includeSubDomains.
X-XSS-Protection	Enables the browser's XSS filter to detect and block reflected XSS attacks.	X-XSS-Protection	X-XSS-Protection: 1; mode=block	- Enables browser XSS protection but shouldn't be relied upon solely. - Combine with CSP for stronger XSS mitigation.
Referrer-Policy	Controls the amount of information in the Referer header sent with requests, improving privacy and preventing data leakage across origins.	Referrer-Policy	Referrer-Policy: no-referrer or Referrer-Policy: strict-origin-when-cross-origin	- Use no-referrer for maximum privacy. - strict-origin-when-cross-origin balances privacy and necessary functionality.
Feature-Policy (Permissions-Policy)	Restricts or enables the use of certain browser features such as geolocation, camera, microphone, or payment information, thus reducing attack surfaces.	Feature-Policy	Feature-Policy: geolocation 'self'; microphone 'none'; payment 'none'	- Tailor the policy based on actual needs. Disable features like camera, microphone, and payment if not needed in e-learning platforms.
Expect-CT	Ensures that the website's SSL/TLS certificates are logged in public Certificate Transparency logs, helping detect certificate mis-issuance.	Expect-CT	Expect-CT: max-age=86400, enforce, report-uri="https://example.com/report"	- Implement if using HTTPS to ensure proper certificate issuance. - Regularly monitor reports for certificate transparency issues.
Cache-Control	Specifies how, and for how long, web content is cached, ensuring that sensitive or dynamic information is not improperly cached by browsers or proxies.	Cache-Control	Cache-Control: no-store, no-cache, must-revalidate	- Use for pages containing sensitive information, such as user authentication or payment data, to avoid caching sensitive data.
Permissions-Policy	Newer alternative to Feature-Policy, controls access to browser features like geolocation, camera, microphone, etc., further restricting the attack surface.	Permissions-Policy	Permissions-Policy: geolocation=(), microphone=(), camera=()	- Specify which features are allowed or disallowed for certain pages, especially those dealing with sensitive data or PII.
Public-Key-Pins (HPKP)	Protects against man-in-the-middle attacks by pinning a site's public key to the browser to ensure the correct certificate is always used.	Public-Key-Pins	Public-Key-Pins: pin-sha256="base64+primarykey"; pin-sha256="base64+backupkey"; max-age=2592000; includeSubDomains; report-uri="/hpkp-report"	- Not commonly used today due to operational risks, but can add security against forged certificates. Use cautiously and with proper backup keys.
Cross-Origin Resource Sharing (CORS)	Controls how resources on a web page are shared across origins, protecting against unauthorized access from external domains.	Access-Control-Allow-Origin	Access-Control-Allow-Origin: https://trusted-domain.com	- Ensure CORS is used carefully to only allow trusted domains, especially if integrating third-party services in an e-learning platform.

Table 2: Detailed description of various Security Headers

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 8s
***Article Received:** 25 May 2023 **Revised:** 12 June 2023 **Accepted:** 30 July 2023*

2. Research Methodology

- Published research papers and articles on web security and security headers.
- Security reports and guidelines from organizations such as OWASP.

2.1 Tools Used:

- **Mozilla Observatory:** This tool assesses and rates websites based on how well they follow security, such as the implementation of security headers.
- **SecurityHeaders.io:** Another tool that reviews a website's security headers and generates a comprehensive report on their settings.
- **Qualys SSL Labs:** This tool evaluates the robustness of HTTPS implementations, especially focusing on HSTS usage.
- **Crawler Scripts:** Automated scripts were deployed to collect real-time information from various e-learning platforms.

2.2 Platforms Analyzed:

- **Moodle** (Example Instance: moodle.org)
- **Blackboard** (Example Instance: blackboard.com)
- **Coursera** (Example Instance: coursera.org)
- **Canvas** (Example Instance: instructure.com/canvas)
- **edX** (Example Instance: edx.org)

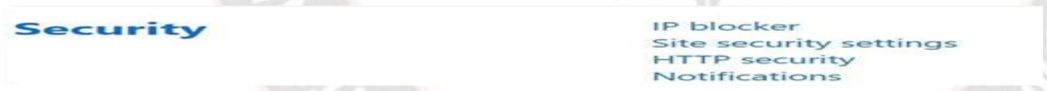
2.3 Data Points Collected:

- Status of key security header implementations, including X-XSS-Protection, CSP, X- Content-Type-Options, HSTS, X-Frame-Options, and Referrer-Policy.
- Ratings or scores given by the tools utilized.
- Effectiveness analysis of security headers in light of recent vulnerabilities highlighted in security advisories.

3. Implementation Work and Analysis

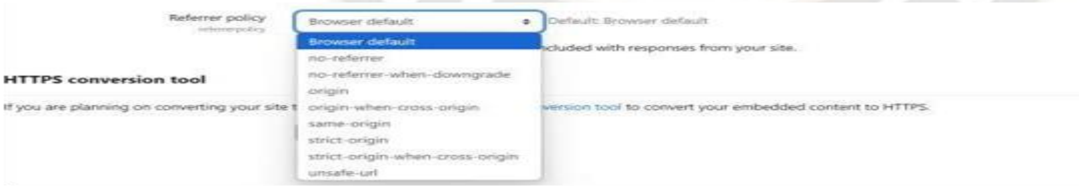
E-learning platforms, require robust security measures to protect user data and system integrity as critical digital infrastructures with security of intellectual property.

Various security headers are inbuilt in various E-Learning platforms. Some of the security headers need to be configured in E-Learning platforms. Plugins play very important role here and are widely used in platforms for information security at client side as well as server side.


Security

IP blocker
Site security settings
HTTP security
Notifications

Fig. 1 Security Mechanisms Applied by Moodle Platform

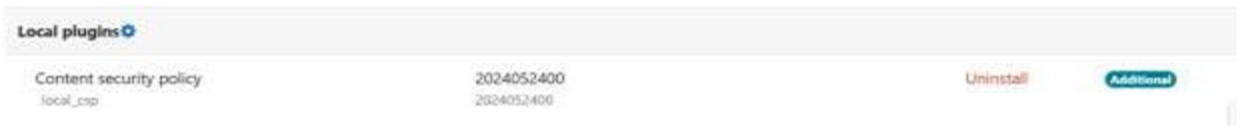

Referrer policy
referrerpolicy
Browser default
Default: Browser default
…cluded with responses from your site.

Browser default
no-referrer
no-referrer-when-downgrade
origin
origin-when-cross-origin
same-origin
strict-origin
strict-origin-when-cross-origin
unsafe-url

HTTPS conversion tool

If you are planning on converting your site t… …version tool to convert your embedded content to HTTPS.

Fig. 2 Referrer Policy in Moodle Platform


Local plugins

Content security policy local_csp	2024052400 2024052400	Uninstall	Additional

Fig. 3 CSP plugin in Moodle

1.1 Results

There are websites available on which one can check their websites for security headers implementation. The online websites scan

the E-Learning websites and display the results stating which security headers are needed to be implemented on the scanned website. Not all websites scan for all security headers. They scan for some most commonly used security headers which are required to be configured. The table given below summarizes the name of website and the security headers for which those websites scan. Table also contains the results those websites produced. The websites used are [9][10][11][12][13][14][15][16]:-
Recommendations for Improving Security Header
mplementation\

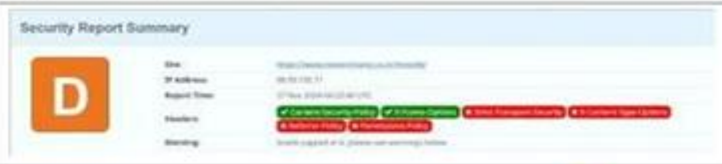
S. N.	Name of Website and Security Headers	Report of Security Scan/Check
1	https://securityheaders.com/ Content-Security-Policy X-Frame-Options Strict-Transport-Security X-Content-Type-Options Referrer-Policy Permissions-Policy	
2	https://www.serpworx.com/check-security-headers X Frame Options X XSS Protection X Content Type Options X Permitted Cross Domain Policies Strict Transport Security Content Security Policy Referrer Policy Feature Policy Expect CT	
3	https://domsignal.com/test/xw39yefx07jklrp5938dwyxxprkbe94g Strict-Transport-Security X-Frame-Options X-Content-Type-Options Content-Security-Policy X-Permitted-Cross-Domain-Policies Referrer-Policy Clear-Site-Data Cross-Origin-Embedder-Policy Cross-Origin-Opener-Policy Cross-Origin-Resource-Policy Cache-Control	
4	https://www.atatus.com/tools/security-header Content-Security-Policy X-Frame-Options Strict-Transport-Security X-Content-Type-Options Referrer-Policy Permissions-Policy	
5	https://developer.mozilla.org/en-US/observatory/ Content Security Policy (CSP) Cookies Cross Origin Resource Sharing (CORS) Redirection Referrer Policy Strict Transport Security (HSTS) Subresource Integrity X-Content-Type-Options X-Frame-Options Cross Origin Resource Policy	 HTTP Observatory Report Scan summary D Scan results

Table 3: Free websites to check for Security Headers and their outcome

For enhancing the security of e-learning platforms, the following recommendations are proposed:

1. **Implement Essential Security Headers:** Ensure that all core security headers, including HSTS, CSP, and X-Frame-Options, are implemented correctly.

2. **Adopt Advanced Headers:** Consider using more advanced headers like Referrer- Policy, Feature-Policy, and Permissions-Policy to further strengthen security.

3. **Regularly Update and Test:** Keep security headers up-to-date with the latest standards and regularly

test their effectiveness.

4. **Utilize Security Headers Scanners:** Employ automated tools to identify and address misconfigurations.

5. **Collaborate with Security Experts:** Seek guidance from security professionals to ensure best practices are followed.

6. **Stay Informed:** Keep abreast of the latest security threats and vulnerabilities to proactively address potential risks.

4. Conclusion

The security of e-learning platforms is paramount to protect sensitive user data and maintain the integrity of the learning environment. By implementing robust security headers and following best practices, e-learning platforms can significantly mitigate security risks and provide a safer online learning experience.

References

1. Akacha, S.A.-L.; Awad, A.I. Enhancing Security and Sustainability of e-Learning Software Systems: A Comprehensive Vulnerability Analysis and Recommendations for Stakeholders. Sustainability 2023, 15, 14132. https://www.mdpi.com/2071-1050/15/19/14132.
2. Shubham Agarwal, Ben Stock, First, Do No Harm: Studying the manipulation of security headers in browser extensions, Network and Distributed Systems Security (NDSS) Symposium 2021 21-24 February 2021, San Diego, CA, USA ISBN 1-891562-66-5, https://dx.doi.org/10.14722/madweb.2021.23016
3. Mlyatu, M.M. and Sanga, C. (2023) Secure Web Application Technologies Implementation through Hardening Security Headers Using Automated Threat Modelling Techniques. Journal of Information Security, 14, 1-15. https://doi.org/10.4236/jis.2023.141001.
4. William J. Buchanan , Scott Helme , Alan Woodward, Analysis of the adoption of security headers in HTTP, IET Information Security, ISSN 1751-8709, doi: 10.1049/iet- ifs.2016.0621.
5. Sudarsanan Nair, S.; Mariappan, K. A Secure Framework for Communication and Data Processing in Web Applications. Eng. Proc. 2023, 59, 1. https://doi.org/ 10.3390/engproc2023059001
6. Artūrs Lavrenovs, F. Jesús Rubio Melón, HTTP Security Headers Analysis of Top One Million Websites, 2018 10th International Conference on Cyber Conflict.
7. Koray Emre KISA, Emin İslam TATLI, Analysis of HTTP Security Headers in Turkey, INTERNATIONAL JOURNAL OF INFORMATION SECURITY SCIENCE K.E. Kısa et al. ,Vol. 5, No. 4
8. Abner Mendoza, Phakpoom Chinprutthiwong, and Guofei Gu. 2018. Uncovering HTTP Header Inconsistencies and the Impact on Desktop/Mobile Websites. In WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186091.
9. https://securityheaders.com/
10. https://www.serpworx.com/check-security-headers
11. https://domsignal.com/test/xw39yefx07jklrp5938dwyxxprkbe94g
12. https://www.atatus.com/tools/security-header
13. https://developer.mozilla.org/en-US/observatory/
14. https://www.hardenize.com/report/researchsaroj.co.in/1729313326
15. https://securityscan.getastra.com/
16. https://www.immuniweb.com/websec/