Large Scale Evaluation of Secure Headers in Wild

Sanggu Han
139@kaist.ac.kr
KAIST
Daejeon, South Korea

Yeongbin Hwang
kyh3565800@kaist.ac.kr
KAIST
Daejeon, South Korea

ABSTRACT

Web applications have become essential to modern life, with billions of people sharing their daily experiences via social network services and hundreds of millions of orders made through e-commerce services every day. However, the prevalence and sophistication of XSS attacks has also increased, posing a significant threat to web security. XSS attacks exploit vulnerabilities in web applications to inject malicious code, which can lead to the theft of sensitive information, the hijacking of user accounts, and the distribution of malware. To address this threat, web developers can use secure response headers, such as the "X-XSS-Protection" and "Content-Security-Policy" headers, to help prevent XSS attacks. In this paper, we evaluate the large-scale deployment of these response headers and assess their effectiveness in preventing XSS attacks. Our results provide insight into the current state of secure header deployment and highlight potential areas for improvement.

1 INTRODUCTION

Web applications have become essential to people's life. Billions of people share their daily life via social network services, and hundreds of millions of orders are made through e-commerce services within a day. In recent years, XSS attacks have become increasingly prevalent and sophisticated, posing a significant threat to web security. XSS attacks exploit vulnerabilities in web applications to inject malicious code into a website, which can then be executed by unsuspecting users. This can lead to a range of security issues, including the theft of sensitive information, the hijacking of user accounts, and the distribution of malware. This characteristic makes the web applications a sweet spot for attackers, and thus over 1,500 XSS CVE were registered this year [4].

To address this threat, web developers can use secure response headers to help prevent XSS attacks. These headers, such as the "X-XSS-Protection" and "Content-Security-Policy" headers, provide instructions to the user's web browser on how to handle potential XSS attacks. By implementing these headers correctly, developers can help protect their users from XSS attacks and improve the overall security of their web applications.

In this paper, we evaluate the large-scale deployment of these response headers in HTTP. We analyze a large dataset of HTTP responses to determine the deployment rates of secure headers and assess their effectiveness in preventing XSS attacks. Our results provide insight into the current state of secure header deployment and highlight potential areas for improvement.

Additionally, we discuss the challenges and limitations of our study, as well as future directions for research in this area. Overall, our work contributes to a better understanding of the role of response headers in preventing XSS attacks and provides valuable information for web developers and security professionals.

2 BACKGROUND

2.1 Cross-Site Scripting (XSS)

XSS is a vulnerability targeted at clients that exploits a vulnerability in a server to send a malicious script to the client. It usually occurs because the server does not properly process the user's input data.

2.2 Defenses for XSS

There are several other methods that web developers can use to prevent XSS attacks. A detailed description of the secure header will be covered in 2.3.

2.2.1 Input sanitization and validation. This method checks user input for potentially malicious code and ensures that only valid input is accepted. For example, suppose a web form only allows users to enter their names. In that case, the web application should validate that the input only contains letters and spaces and reject any input that contains special characters or numbers. In addition, if user input contains special characters such as "<" and ">," which are often used in HTML tags, these characters should be removed or escaped.

2.2.2 Context-aware output encoding. This method uses the appropriate encoding for the context in which the user input is displayed. For example, if user input is displayed in HTML, it should be HTML-encoded to prevent any potential XSS attacks. This ensures that any potentially dangerous characters are rendered as harmless text rather than being executed as code.

2.3 Secure Header

Secure headers are HTTP response headers that provide the browser with additional instructions on how to handle incoming requests and responses.

2.3.1 Content-Security-Policy. The Content-Security-Policy (CSP) is an HTTP header that tells the user's web browser how to handle possible XSS (cross-site scripting) attacks. The CSP header allows web developers to specify which sources are allowed to load resources on the page. For instance, the header can be used to say that only resources from a certain domain can be loaded, or that certain types of resources, like inline JavaScript or third-party fonts, can't be loaded.

2.3.2 X-XSS-Protection. The X-XSS-Protection header allows web developers to enable the browser's built-in XSS protection. By setting the header to "1; mode=block", developers can instruct the browser to prevent the rendering of a page if it detects a potential XSS attack. This can help protect users from XSS attacks and improve the overall security of the web application.

2.3.3 X-Content-Type-Options. X-Content-Type-Options is an HTTP response header that can be used to indicate to the browser whether or not it should try to detect the MIME type of a file that's being downloaded. The only allowed value for this header is `nosniff`, which tells the browser not to try to guess the MIME type of the file and to use the one that's provided by the server. This can help prevent XSS attacks, by ensuring that the browser only executes files with the correct MIME type.

3 METHODOLOGY

3.1 Definition of Target Secure Headers

To defend against XSS attacks, one approach is to use HTTP response headers. These headers instruct the browser on how to handle incoming requests and responses and can be utilized to avoid the execution of malicious code. There are several headers that can be used for this purpose, and Mozilla and the OWASP Secure Headers Project provide recommendations on which headers to deploy.

- Mozilla is a non-profit organization maintaining the widely used web browser Firefox. Mozilla maintains a list of suggested HTTP response headers [5] that may be used to improve the security of a website as part of its commitment to online security.
- The OWASP Secure Headers Project is a community-driven effort to promote the usage of secure headers to improve web application security. Additionally, the project maintains a list of suggested headers [6] that may be utilized to prevent a variety of attacks, including XSS attacks.

In this paper, we will focus on the fundamental cause of XSS attacks: the execution of JavaScript in the browser. We will analyze the headers recommended by Mozilla and the OWASP Secure Headers Project, and select the most effective headers for our use case.

Guideline The guideline is a set of principles for selecting secure headers that focus on preventing the execution of malicious scripts or restricting the coverage of JavaScript on a website. This is a proactive approach to web security that aims to prevent potential attacks by limiting the capabilities of potentially harmful scripts.

The guideline states that the selected secure headers should focus on addressing the fundamental causes of security issues. This means that the selected headers should be specifically designed to block the execution of malicious scripts or only allow the specific scripts that have been predefined as part of the server's request. Additionally, the guideline includes the restriction of the coverage that JavaScript can obtain on the website. This can help prevent attackers from using JavaScript to obtain to sensitive information or perform unauthorized actions. By limiting the scope of JavaScript, the website can be made more secure and less vulnerable to attacks. Some examples of secure headers that could be selected using the guideline are X-XSS-Protection and Content-Security-Policy.

The guideline does not include provisions for managing cookies or requiring HTTPS, as these are not directly related to the prevention of malicious script execution. However, these measures can also improve the security of a website and should be considered as part of a comprehensive security strategy.

Table 1: selected secure headers

Secure headers	Type
Cross-Origin-Embedder-Policy	Restriction
Cross-Origin-Opener-Policy	Restriction
Cross-Origin-Resource-Policy	Restriction
Content-Security-Policy	Execution
Feature-Policy	Execution
X-Content-Type-Options	Execution
X-Frame-Options	Execution
X-XSS-Protection	Execution

Table 1 shows a selection of secure headers that can be used to improve the security of a website. These headers are selected based on the guideline.

3.2 Crawling Secure Headers

Current secure headers We implemented a web crawler that can extract secure headers from the response headers of websites, and for that, we collected the target websites from the Alexa Top 1M websites [1]. The Alexa Top 1M websites are a list of the most popular websites on the internet. However, not all of the websites in the Alexa Top 1M list are valid or still active. Some of these websites may be malicious, and some may have disappeared or been taken offline. This means that it is not possible to get a complete and accurate list of response headers from the Alexa Top 1M websites.

Past secure headers We collected past response headers for comparison with historical data. The data is obtained from a website called crawler.ninja [2], which is a site that provides raw data and statistics about response headers.

3.3 Comparison of Secure Headers

The analysis of the data collected from the Alexa Top 1M websites can provide valuable insights into the use of secure headers on the web. By comparing the values of secure headers obtained from the past with newly crawled values, we can understand how the number of uses of each header has increased over time, and identify any newly introduced values.

This information can be used to understand how the use of secure headers is evolving and can provide insights into the ways that websites are using these headers to protect themselves from potential attacks. For example, we might find that the use of certain headers has increased significantly, indicating that they are becoming more popular and effective for protecting websites.

Additionally, the analysis of weak field options can help us identify websites that are vulnerable. This can be useful for identifying potential vulnerabilities and making recommendations for improving the security of these websites.

3.4 Investigation on Supported Headers for Each Browser

We also investigated the websites that may be vulnerable due to a mismatch between a website's request for a particular header and the support for that header in different web browsers. To do

this, we collected the supported headers for each web browser and looked for instances where a website has requested a header that is not supported by one or more browsers. For example, if a website requests a header that is not supported by a particular browser, the browser will not be able to enforce the security policies specified in that header. This can allow attackers to bypass the security measures put in place by the website, and potentially gain access to sensitive information or perform unauthorized actions.

To investigate the supported headers for each web browser, we are using the caniuse API in node js. The caniuse API provides detailed information about the support for response headers in different web browsers, including secure headers. By using the caniuse API, we can easily access information about the supported headers for each browser and use this information to identify potential vulnerabilities.

4 EVALUATION

We evaluated websites' secure header usage and tracked trends shifting over 6 months. The focus was on the following statistics:

(1) The overall adoption rate for secure headers related to script execution (§4.2)
(2) Rules of each secure header (§4.3) and misused cases (§??)

4.1 Experimental Setup

We conducted our evaluation using two large datasets. The first dataset was provided by crawler.ninja [2], which regularly publishes reports of crawling Alexa's top 1 million domains [1] along with statistical data. The dataset we used was from June and contained 820,337 response headers. The second dataset was obtained by crawling the same list of domains in December and contained 650,162 response headers. From this dataset, we extracted data on the usage of header fields, rules for secure headers, and headers using vulnerable secure header combinations.

4.2 Overall Adoption Trend

To observe the overall trend of secure header adoption in the web nature, we first analyzed the usage of each secure header by calculating the proportion of headers that included a secure header. In figure 1, 2 and table 2, our analysis of the data shows a clear trend of increasing adoption rates for relatively older secure headers. We can tell that these headers have successfully gained acceptance among web developers. On the other hand, the adoption rate for recently proposed secure headers does not show the same upward trend. This may be due to a lack of awareness or understanding of these newer headers among web developers. In order to increase the adoption of these newer headers, it may be necessary to implement targeted marketing and educational initiatives to raise awareness and understanding among web developers.

4.3 Rules of Secure Headers

We will investigate the use of rules for secure header implementation to enhance the system's overall security.

4.3.1 Content-Security-Policy. To analyze the usage of the content-security-policy header, we first extracted the unique rules from the collection of content-security-policy rules that were gathered.

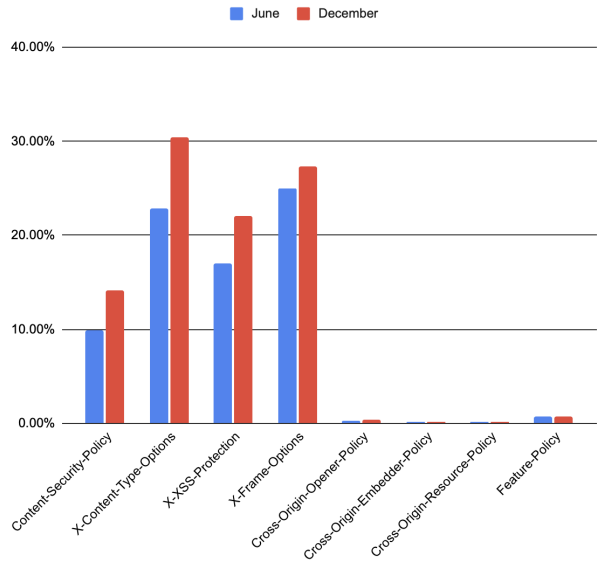

Figure 1: Graph of overall secure header adoption.

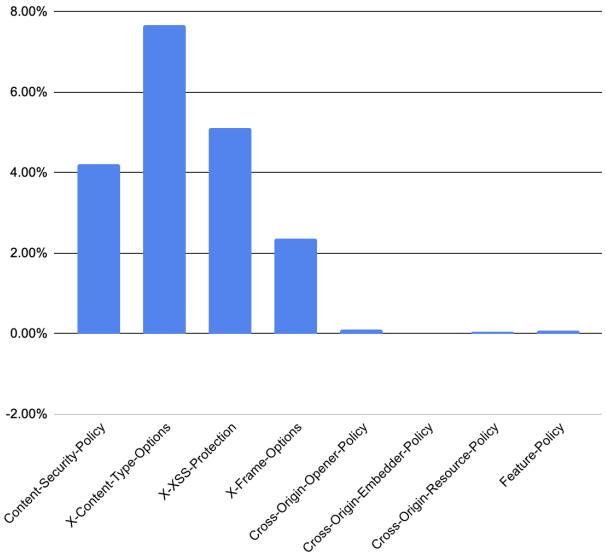

Figure 2: Increase rate of each secure header over 6 months in percentages.

We then analyzed the occurrence of each directive in those unique rules. As we can see in figure 3, the most frequently used directive is *script-src*, which indicates that web developers employ this header to prevent cross-site scripting attacks mostly.

The third-most commonly used directive in data is *frame-ancestors*, which improves protection against clickjacking attacks in a similar way to the "X-Frame-Options" directive. In December, approximately 178,000 headers utilized this field to protect their services from clickjacking attacks. If both "frame-ancestors" in the "Content-Security-Policy" and "X-Frame-Options" is set, the "frame-ancestors" directive takes precedence. We have observed that over 186,000

Table 2: Usage of each secure headers in June and December respectively. Percentages next to numbers are calculated by dividing total number of headers in dataset, which are 820,337 in June and 650,162 in December.

Secure headers	June	December
Cross-Origin-Embedder-Policy	837 (0.10%)	662 (0.10%)
Cross-Origin-Opener-Policy	2,504 (0.31%)	2,608 (0.40%)
Cross-Origin-Resource-Policy	1,307 (0.16%)	1,309 (0.20%)
Content-Security-Policy	81,178 (9.90%)	91,751 (14.11%)
Feature-Policy	5,620 (0.69%)	4,847 (0.75%)
X-Content-Type-Options	187,114 (22.81%)	198,052 (30.46%)
X-Frame-Options	205,155 (25.01%)	177,939 (27.37%)
X-XSS-Protection	138,994 (16.94%)	143,378 (22.05%)

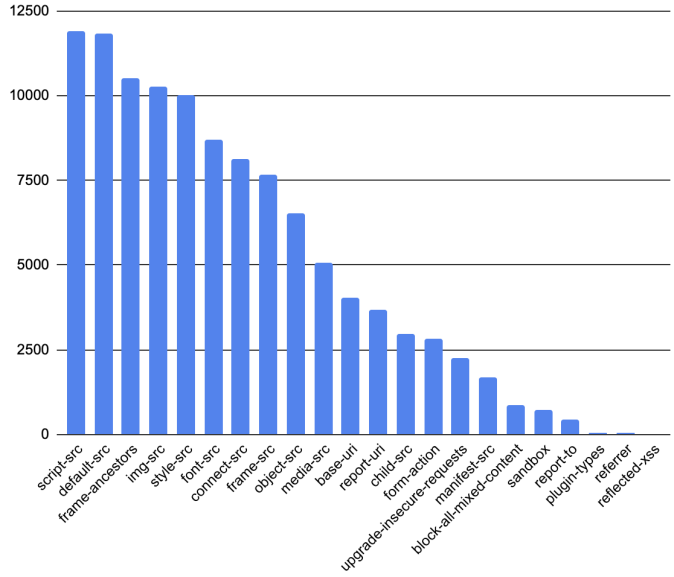

Figure 3: Directive usage in unique rules of Content-Security-Policy

cases have set at least one of these headers to prevent clickjacking attacks

Besides this, *frame-ancestors* is ranked as the third-most used directive in data. It is used to specify which websites are allowed to embed in a <frame>, <iframe>, or <object> element. This helps to prevent web page from being embedded in a malicious website, which could be used to clickjacking. This directive is more powerful and flexible way of controlling the use of <frame>, <iframe>, or <object> element than X-Frame-Options, and it has obsoleted the X-Frame-Options header.

4.3.2 X-Frame-Options. The x-frame-options header is used to indicate whether a web page should be allowed to be displayed in a <frame>, <iframe>, or <object> element on another web page. This secure header also protects against clickjacking like *frame-ancestors* directive in Content-Security-Policy, but in a less granular way. In table 2, we can see that about 178,000 headers adopted this field to protect their services from clickjacking attack this December. With the usage of *frame-ancestors* combined, there are more than 186k cases that try to prevent clickjacking via secure header have been observed.

4.4 Case Study: X-XSS-Protection and Content-Security-Policy

The X-XSS-Protection header is a security feature that was commonly used by web browsers to prevent XSS attacks. However, this header has been deprecated and is no longer supported by most modern browsers. It is strongly recommended that do not use this header alone, as it provides only limited protection against XSS attacks. Instead, developers should use a Content-Security-Policy header that includes a *script-src* directive to specify which sources are allowed to load scripts to provide more comprehensive protection against XSS attacks and other types of malicious code injection.

We discovered that over 87,000 domains are using X-XSS-Protection headers without the *script-src* directive, making them vulnerable to XSS despite their attempts at protection. This indicates that developers should stay up-to-date with secure header trends in order to provide proper web security.

5 FUTURE WORK

In order to further advance the research presented in this paper, there are two potential avenues for future investigation. The first is to analyze the secure headers that were not included in this study, as they are not related to script execution. While this paper focuses on the headers related to script execution, there is still a significant body of knowledge to be gained from studying the other headers and their impact on website security.

The second potential avenue for future work is the development of an automatic static analysis tool for secure headers. While the rules for implementing secure headers are relatively simple, there is always the possibility of human error, which can lead to vulnerabilities in websites. There are already tools available for analyzing specific secure headers such as Content-Security-Policy [3]. However, these tools do not provide a comprehensive analysis of all the headers, and they may not always be effective in detecting vulnerabilities. By developing a more comprehensive and robust automatic analysis tool, we can improve upon the existing solutions and provide a more effective way to ensure the security of websites.

Overall, by investigating these two areas, we can expand upon the findings of this paper and provide a more comprehensive understanding of the role of secure headers in website security.

6 CONCLUSION

It is important to note that while secure headers are an important tool for improving the security of the web, many developers may misuse them or fail to implement them properly. For example, some developers may use secure headers in an attempt to block legitimate requests, or may not configure them correctly, leading to vulnerabilities in their web applications. In order to ensure that secure headers are used effectively and properly, it is important for web developers to understand how to use them correctly and to follow best practices for their implementation. Additionally, regular testing and review of web applications can help to identify and fix any issues with the use of secure headers.

Our extensive evaluation of headers has revealed that the adoption of secure headers can greatly improve web security. However, it is important to note that the misuse of these headers can lead to vulnerabilities. In light of this, it is essential for web developers to make more widespread use of secure headers in order to keep their websites and users safe.

REFERENCES

[1] Alexa [n. d.]. Alexa Top 1M Websites. https://www.expireddomains.net/alexa-top-websites/.
[2] Crawler-Ninja [n. d.]. Crawler.Ninja. https://crawler.ninja/.
[3] csp-evaluator [n. d.]. CSP Evaluator. https://csp-evaluator.withgoogle.com.
[4] CVEXSS [n. d.]. Search result of XSS in CVE. https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=xss.
[5] Mozila [n. d.]. Mozila HTTP Secure Header. https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers.
[6] OWASP [n. d.]. OWASP Secure Headers Project. https://owasp.org/www-project-secure-headers/.