Analyzing the Feasibility of Adopting Google's Nonce-Based CSP Solutions on Websites

1st Mengxia Ren
Department of Computer Science
Colorado School of Mines
Golden, USA
mengxiaren@mines.edu

2nd Anhao Xiang
Department of Computer Science
Colorado School of Mines
Golden, USA
xianganhao@mines.edu

3rd Chuan Yue
Department of Computer Science
Colorado School of Mines
Golden, USA
chuanyue@mines.edu

2025 IEEE/ACM 47th International Conference on Software Engineering (ICSE) | 979-8-3315-0569-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICSE55347.2025.00193

***Abstract*—Content Security Policy (CSP) is a leading security mechanism for mitigating content injection attacks such as Cross-Site Scripting (XSS). Nevertheless, despite efforts from academia and industry, CSP policies (in short, CSPs) are not widely deployed on websites, and deployed CSPs often have security issues or errors. Such low and insecure CSP deployment problems are mainly due to the complexity of the CSP mechanism. Google recently proposed four *nonce-based CSP solutions* which are simpler and more secure compared to traditional whitelisting-based CSP solutions. Google successfully deployed their nonce-based CSP solutions on over 160 services, covering 62% of all outgoing Google traffic. These nonce-based CSP solutions use simple CSPs but provide fine-grained control of web resources; therefore, if widely adopted on many other websites, they can be very helpful on addressing the low and insecure CSP deployment problems. In this paper, we evaluate the feasibility of adopting Google's nonce-based CSP solutions on the Tranco top 10K websites. We construct a crawling tool to automatically visit websites, simulate user interactions, and insert four CSPs to collect the CSP violations triggered under them. We investigate the adoptability of the nonce-based CSP solutions, adoption issues, and the stability of adopting them on websites by analyzing the CSP violations triggered under the inserted CSPs. We found that most websites can adopt the nonce-based CSP solutions on all their webpages visited in our study. For websites that cannot, usually the adoption is hard on around 40% of their webpages. Overall, our results are very encouraging and can be helpful in promoting the proper deployment of CSPs on many websites.**

I. Introduction

The extensive functionality of the Web makes it vulnerable to content injection attacks, most prominently Cross-Site Scripting (XSS) [5], [15], [16], that exploit vulnerabilities in web applications to insert malicious code to be executed on users' browsers. To secure the Web, major browsers have adopted Content Security Policy (CSP) which is a leading mechanism for protecting webpages against content injection attacks [9], [13], [28]. Web developers can deploy CSP policies (in short, CSPs) to allow permissible resources (e.g., scripts including JavaScript code or files) to be loaded or executed and behaviors (e.g., form submissions) to occur on their websites. There are two CSP deployment modes: ***enforcement*** and ***report-only***; resources violating CSPs deployed under the enforcement mode will be blocked, whereas resources violating CSPs deployed under the report-only mode will not be blocked but will be reported as CSP violations [28].

A CSP is composed of directives, and each directive is typically composed of a directive name with a set of directive values [9], [13], [28]. A directive name specifies a certain type of resources to be controlled, while directive values specify the allowed resources or behaviors on a webpage. Different types of resources (e.g., frames, scripts, and fonts) are controlled by the corresponding directives. For example, the "script-src https://script.com" directive is composed of a *script-src* directive name with a script source URL (i.e., *https://script.com*) directive value. This directive only allows scripts from *https://script.com* to be loaded or executed on a webpage. CSPs can be deployed on webpages via HTTP(S) response headers or via HTML <meta> tags [9], [13], [28].

While CSP offers a promising defense against content injection attacks, it is not widely deployed on websites yet (e.g., they appeared only on 12.06% websites as reported in a 2020 study [24], and this number increased to 17.74% websites as reported in a 2023 study [20]), and most deployed CSPs have security issues or errors due to the complexity of the CSP mechanism [3], [20], [21], [24], [33]. Web developers often struggle with the proper (i.e., correct and secure) CSP deployment due to technical, application, knowledge gap, and cost-related roadblocks [25]. These issues have still not been solved, despite efforts from researchers and organizations.

Google has a long history in both developing and deploying CSPs to protect their services against content injection attacks [31]–[33]. Recently, they proposed four nonce-based CSP solutions with different security levels to balance security and adoption efforts [32]. These nonce-based CSP solutions use unique nonces or hash values to control scripts without requiring maintaining whitelists of resource sources. They can help address CSP deployment issues as they are simpler and more secure compared to traditional whitelisting-based CSP solutions [32], [33]. They are simpler especially because nonce-based CSPs themselves are short and literally fixed policies except that nonces should be unique for each HTTP(S) response and should be unguessable by attackers [28], [33]. Furthermore, they can allow resources that are requested by already trusted resources to be loaded or executed without matched nonces or hash values, which provides a way to allow dynamic resources and third-party resources [32], [33]. They are more secure especially because they directly provide

1558-1225/25/$31.00 ©2025 IEEE
DOI 10.1109/ICSE55347.2025.00193

fine-grained control of web resources on each webpage based on unique nonces or hash values. Therefore, they can be considered as page-level CSP deployment solutions that adhere to the principle of least privilege to load web resources on a webpage. With unique nonces, they also directly address the problem that in practice webpages of a website often share the same CSP (e.g., 90% of all analyzed website origins that deployed CSPs share the same CSP on their webpages [4]).

According to the coverage ranges of XSS attack sinks [32], Google's nonce-based CSP solutions at levels 1 and 2 (as detailed in Section II-B and Table I) are referred to as ***weaker nonce-based CSP solutions***, while the other two at levels 3 and 4 are referred to as ***stronger nonce-based CSP solutions***. Google successfully deployed their nonce-based CSP solutions on over 160 services, covering 62% of all outgoing Google traffic [32]. This resulted in blocking over 60% of XSS attacks in total, with a notable blockage of 80% of XSS attacks on highly sensitive domains [32].

In this paper, we focus on analyzing the feasibility of adopting Google's nonce-based CSP solutions on websites. Specifically, we answer the following three research questions under the assumption that websites want to properly deploy CSPs on their webpages. ***RQ1***: To what extent could websites adopt Google's nonce-based CSP solutions? ***RQ2***: What are the issues encountered by websites that would fail to adopt Google's nonce-based CSP solutions, and how could web developers address them? ***RQ3***: How stable could the adoption of Google's nonce-based CSP solutions be on websites across different crawlings? We implemented a crawling tool to automatically visit up to 101 webpages (including one landing page and 100 subpages) of each of the Tranco [18] top 10K websites, simulate user interactions, insert four CSPs (which can compose the two stronger nonce-based CSP solutions), and collect the corresponding CSP violations. We analyze the feasibility of adopting Google's nonce-based CSP solutions on websites based on CSP violations generated under the inserted CSPs. To analyze the stability of adopting the nonce-based CSP solutions on websites, we performed the website crawling three times and compared their results. We collected 7,034, 6,895, and 7,427 unique websites in the first, second, and third crawlings, respectively.

Overall, we found from the first crawling that 62.74% of websites can adopt Google's nonce-based CSP solutions on all their webpages visited in our study, and 21.53% of websites can even adopt the stronger ones on all their webpages. Meanwhile, for websites that cannot adopt the nonce-based CSP solutions on all their webpages visited in our study, usually the adoption is hard on around 40% of their webpages. We further analyzed the adoption issues for webpages that cannot adopt any nonce-based CSP solutions or the stronger ones. It is worth noting that if websites can address the adoption issues of nonce-based CSP solutions caused by first-party resources (which is typically doable because first-party resources are under the full control of each website and changing their uses does not require the cooperation from third parties), the number of websites that can adopt the nonce-based CSP solutions and the stronger ones will increase by 13.39% and 22.64%, respectively. Additionally, across three crawlings, the percentages of websites that can adopt nonce-based CSP solutions (62.74%, 63.71%, and 63.16%, respectively) are similar. In total, 5,797 websites were commonly visited across three crawlings, among which 59.56% can always adopt the nonce-based CSP solutions on all their webpages visited in our study, while 34.91% always cannot adopt the nonce-based CSP solutions on all their webpages visited in our study.

We make three major contributions in this paper:

- We design a crawling tool to directly insert four CSPs, which can compose Google's two stronger nonce-based CSP solutions, under the report-only mode to each visited webpage to collect corresponding CSP violations. This provides a straightforward way to analyze the feasibility of adopting the nonce-based CSP solutions on websites.
- We perform a large-scale measurement study on 10K websites and analyze the feasibility of adopting the nonce-based CSP solutions on their webpages to help promote the proper deployment of CSPs.
- We obtained very encouraging results. We found that most websites can adopt the nonce-based CSP solutions on all their webpages visited in our study. Meanwhile, a noticeable percentage of websites will be able to adopt the nonce-based CSP solutions on all their webpages if they move foward to address the adoption issues caused by their first-party resources. The adoption of nonce-based CSP solutions is stable across the three crawlings.

II. Background and Related Work

A. Background of CSP

CSP Directives. Developers can whitelist resource sources or use nonces or hash values as directive values in the corresponding CSP directives to control web resources on webpages. Resources from whitelisted sources or with matched nonces or hash values are allowed to be loaded or executed. However, CSP also provides the following ***unsafe directive values*** which can cause security issues: (1) the " 'unsafe-inline' " directive value allows any scripts and event handlers included in a webpage to be loaded or executed, (2) the " 'unsafe-eval' " directive value allows the execution of any eval-like functions, (3) the " 'wasm-unsafe-eval' " directive value allows the execution of any wasm eval functions, (4) the "*" value allows resources from any sources with an "https:" or "http:" scheme to be loaded or executed, and (5) scheme values (e.g., a "data:" value) allow the execution of scripts from any sources with the corresponding scheme values. Browsers will ignore the " 'unsafe-inline' " value when it and a nonce or hash value both appear in a CSP. Weichselbaum et al. [33] proposed the " 'strict-dynamic' " directive value which was later standardized in the CSP Version 3 specification [28] to allow scripts with matched nonce or hash values to further dynamically include non-parser-inserted scripts (i.e., inserted by using the document.createElement('script') API controlled by the JavaScript runtime, not by using the document.write()

API controlled by the HTML parser [28]) into first-party websites. Browsers will ignore the whitelisted resource sources when the “ ‘strict-dynamic’ ” value also appears in a CSP.

CSP Violations. A CSP violation will be triggered on a browser when a resource access (e.g., retrieving a JavaScript file) or behavior (e.g., form submission) violates the CSP deployed on a webpage. CSP violation reports can be sent to the developer-specified hosts or servers via reporting directives (i.e., a “report-uri” or “report-to” directive); they include information of violating resources such as their source files, their types, and the CSP directives they violated.

CSP Deployment Modes and Ways. CSP has two deployment modes: *enforcement* and *report-only*. In the enforcement mode, browsers will block or disable resources and behaviors that violate the deployed CSPs. In the report-only mode, browsers will only report CSP violations (typically for web developers to test policies), without taking enforcement actions. Developers can deploy CSPs on webpages in two ways: via an HTTP(S) response header or an HTML <meta> tag. A CSP in the enforcement mode can be deployed by developers via a “Content-Security-Policy” HTTP(S) response header or an HTML <meta> tag. However, a CSP in the report-only mode can only be deployed by developers via a “Content-Security-Policy-Report-Only” HTTP(S) response header. In addition, content that precedes an HTML <meta> tag specifying a CSP policy will not be controlled by the policy [28].

B. Related Work

CSP Deployment Problems. Although CSP is a leading mechanism for protecting webpages against content injection attacks, CSP policies are not widely deployed in practice [20], [24], [33], [34], and deployed policies often have security issues or deployment errors [20], [24], [27], [33]. Weichselbaum et al. found that 94.72% of deployed CSPs ineffectively protected webpages against XSS attacks because of policy misconfigurations and insecure whitelist entries [33]. Calzavara et al. found that 92.4% of websites that deployed CSPs were vulnerable to XSS attacks due to the usage of an “ ‘unsafe-eval’ ” or “ ‘unsafe-inline’ ” directive value [2]. Calzavara et al. also found that 90% of all analyzed website origins that deployed CSPs shared the same CSPs on all their webpages, and only 1% of analyzed origins had at least one webpage that deployed a secure CSP [4]. Roth et al. found that CSP has been increasingly used for other purposes (such as frame control and TLS enforcement) in addition to the traditional content restriction purpose, and insecure practices (e.g., using an “ ‘unsafe-eval’ ” or “ ‘unsafe-inline’ ” directive value) were present on 90% of 421 websites that deployed CSPs for script content restriction [24]. Ren et al. found that most deployed CSPs do not sufficiently protect webpages from the directive coverage and secure use perspectives [20].

In a 2023 study, Golinelli et al. specifically investigated the CSP nonce reuse insecure practices [8]. They found that only 4.5% (2,271 out of the Tranco top 50K) websites use a CSP nonce in at least one of their webpages; meanwhile, among those 2,271 websites, 26.3% of them reuse the same nonce value in more than one response (with 93.8% of reuses even across sessions) due to either web cache or server-side code reasons, while 22.1% of them use a CSP nonce shorter than the recommended 128 bits long (before base64-encoding) [8]. They did not analyze what types of nonce-based solutions are deployed on those 2,271 websites, and did not analyze if majority (95.5%) of websites can feasibly deploy nonce-based solutions. We investigate the feasibility of adopting Google’s nonce-based CSP solutions on all visited websites. Therefore, our and their studies differ significantly but complement each other: when websites start to adopt Google’s nonce-based CSP solutions encouraged by our study and supported by our tools, it is important to avoid reusing nonces as highlighted in [8].

Underlying Reasons for CSP Deployment Problems. Several studies investigated the underlying reasons for CSP deployment problems. Roth et al. conducted a qualitative semi-structured interview study involving 12 web developers, and found that factors such as missing web framework support, complexity of the CSP mechanism (thus the high monetary and time cost), and dependence on third-party code could all block a proper CSP deployment [25]. Steffens et al. analyzed resource usage on websites by using in-browser hooks, and found that proper CSP deployment on most first-party websites requires the cooperation from third parties [27]. Meanwhile, these studies opined that event handlers were one major reason for websites to deploy CSPs by using an “ ‘unsafe-inline’ ” directive value instead of using nonce-based CSPs. However, the real scenarios of how nonce-based CSP solutions work on websites may not be revealed by analyzing website resource usage through in-browser hooks (unless all browser APIs in question are comprehensively hooked) or by interviewing a small number of web developers (unless they are experts of nonce-based CSP solutions). Our work analyzes the adoptability of nonce-based CSP solutions, adoption issues, and the stability of adopting them on 10K websites directly based on CSP violations triggered under four inserted CSPs which can compose the two stronger nonce-based CSP solutions.

Automated CSP Generation. To promote CSP adoption, some studies explored automated CSP generation [7], [17]. However, these studies focused on the automated whitelisting-based CSP generation. Compared to traditional whitelisting-based CSP solutions, Google’s nonce-based CSP solutions are simpler and more secure, and they can help address CSP deployment issues (Section I). Our work and findings could benefit future automated CSP generation studies by shedding light on the feasibility of adopting the nonce-based CSP solutions on websites thus the proper deployment of CSPs.

Google’s Nonce-Based CSP Solutions. Recently, Google proposed four ***nonce-based CSP solutions*** as shown in Table I. These four solutions provide protection at different security levels, and all of them do not allow plugins that are embedded in <object>, <embed>, or <applet> tags to be loaded or executed and URLs in <base> tags to be fetched on a webpage. The nonce-based CSP solution at level 4, providing the highest security, only allows scripts matched with nonces or hash values to be loaded or executed. In addition to allowing scripts with matched nonces or hash values, the nonce-based CSP solution at level 3 also allows non-parser-inserted scripts requested by already trusted resources to be loaded or executed without matched nonces or hash values. These two levels of solutions are referred to as ***stronger nonce-based CSP solutions*** since they can cover most XSS sinks [32]. The nonce-based CSP solution at level 2 further allows eval-like functions to be executed on webpages. The nonce-based CSP solution at level 1 further allows event handlers with matched hash values to be loaded or executed on webpages. However, it does not ensure that event handlers execute as the developer intends because attackers can inject their malicious code with the same function name which is used for generating hash values. The nonce-based CSP solutions at levels 1 and 2 are referred to as ***weaker nonce-based CSP solutions*** since they can cover fewer XSS sinks due to the usage of an " 'unsafe-eval' " or " 'unsafe-hashes' " directive value.

TABLE I: Google's Four Nonce-based CSP Solutions.

CSP Solution	***Security Level***	***Security Categorization***
script-src 'nonce-r4nd0m'; object-src 'none'; base-uri 'none';	4	Stronger
script-src 'nonce-r4nd0m' 'strict-dynamic'; object-src 'none'; base-uri 'none';	3	Stronger
script-src 'nonce-r4nd0m' 'strict-dynamic' 'unsafe-eval'; object-src 'none'; base-uri 'none';	2	Weaker
script-src-attr 'unsafe-hashes' 'hash_values'; script-src-elem 'nonce-r4nd0m' 'strict-dynamic' 'unsafe-eval'; object-src 'none'; base-uri 'none';	1	Weaker

Google successfully deployed nonce-based CSP solutions on 160 services, covering 62% of all their outgoing traffic [32]. Nonce-based CSP solutions are simpler and more secure compared to traditional whitelisting-based CSP solutions, and can thus help address the CSP deployment problems (Section I). In this work, we explore the feasibility of adopting them on websites to help promote the proper CSP deployment.

III. Design of Our Study

Our unique goal in this study is to analyze the feasibility of adopting Google's nonce-based CSP solutions on websites. There are two ways to conduct web measurements: utilizing vantage points of web archives (typically using Internet Archive (IA)) to perform an archive-based measurement, or constructing web crawlers to perform a traditional live measurement [11]. However, a large-scale analysis of dynamic web content is difficult when using the IA due to the rate-limiting policy, and IA aggregates data from multiple contributors which may cause a bias in results [11]. To achieve our goal, we use a traditional way to perform a live measurement by constructing a crawling tool to automatically visit webpages for each website, directly insert four CSPs under the report-only mode to each webpage, and collect CSP violations triggered under the inserted CSPs. The feasibility of adopting the nonce-based CSP solutions on websites is analyzed based on the CSP violations triggered under the inserted CSPs. CSP violations directly reflect browsers' policy enforcement and provide a more straightforward way for us to answer our research questions than using other methods such as hooking browser APIs to analyze the webpage content and the allowed vs. disallowed HTTP(S) requests. Our crawling tool performs a full browser (i.e., using a full-fledged browser instead of a headless agent like cURL [1], [12]) and stateless (as in [4]) crawling to simulate a real user's browsing and avoid the impact of the webpage access order on our results.

A. Crawling and Data Collection Framework

Figure 1 illustrates our website crawling and data collection framework. Our crawling tool constructed upon Puppeteer [19] consists of four components: a *CSP inserter*, a *user interaction simulator*, a *request and response collector*, and an *HTML document collector*. It instructs the browser, starting from the landing page of a website, to collect same-site links from webpages and sequentially visit webpages according to the collected links. We recursively repeat the process until we collect up to 101 webpages (one landing page and 100 subpages) or exhaust the webpages to visit for each website. Our crawling tool waits up to 30 seconds to load a webpage. After each webpage is fully loaded, it collects the original CSPs (if they exist) deployed on the webpage, HTTP(S) requests and responses, and the HTML document in 30 seconds.

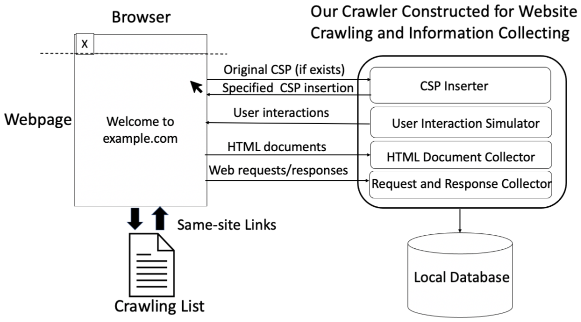
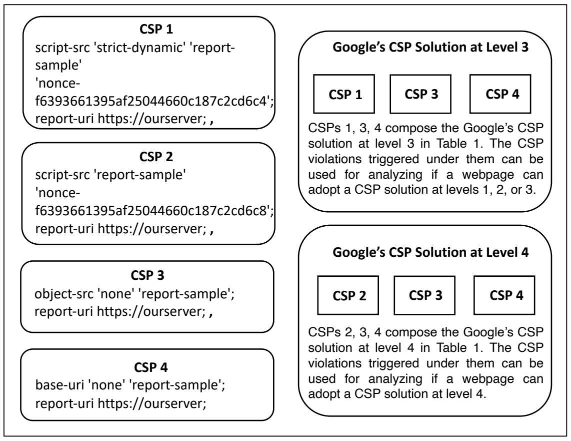


Fig. 1: Website Crawling and Data Collection Framework.

As shown in Figure 2, the CSP inserter component inserts four specified CSPs under the report-only mode to each webpage via a "Content-Security-Policy-Report-Only" HTTP(S) response header with CSPs separated by commas per the guideline on Page 44 of [32] (which follows RFC 2616 [22] and RFC 9110 [23] on using comma-separated field values). CSP 1 allows scripts with an "f6393661395af25044-660c187c2cd6c4" nonce value and non-parser-inserted scripts requested by already trusted scripts to be loaded or executed. CSP 2 only allows scripts with an "f6393661395af25044660c187c2cd6c8" nonce value to be loaded or executed. CSP 3 does not allow plugins embedded in <object>, <embed>, or <applet> tags to be loaded or executed. CSP 4 does not allow URLs in <base> tags to be fetched on a webpage.

Note that when multiple policies are present, each must be enforced (i.e., under the enforcement mode with the 'AND' logic among CSPs) or reported (i.e., under the report-only mode without the interference among CSPs) [29]. CSPs 1, 3, and 4 in Figure 2 do not have conflicting directives; therefore, the 'AND' of their directives directly compose the effective nonce-based CSP solution at level 3 in Table I (ignoring its missing "report-uri" directive and the 'report-sample' directive value for "script-src", and its nonce value difference which is for simplified illustration) under the enforcement mode. Similarly, the 'AND' of the directives of CSPs 2, 3, and 4 directly compose the effective nonce-based CSP solution at level 4 under the enforcement mode. Since these four CSPs are inserted under the report-only mode in our study, they will not block any violating resources, not interfere with the functionality of any original webpages (no harm and no benefit to them), and not interfere with the original CSPs (if they exist, regardless of the enforcement or report-only mode); they will only separately generate CSP violations for the resources without affecting each other. Therefore, analyzing the violations reported by CSPs 1, 3, and 4 together (or by CSPs 2, 3, and 4 together) is equivalent to analyzing the violations that would be reported by the nonce-based CSP solution at level 3 (or at level 4). Analyzing the violations reported by CSPs 1, 3, and 4 together but excluding the violations due to eval-like functions (and further event handlers, etc.) is equivalent to analyzing the violations that would be reported by the nonce-based CSP solution at level 2 (and further at level 1). Each triggered CSP violation contains the detailed information about the violating resource (e.g., type of the resource and its source file) and the violated CSP that can be used in our analysis.



```
script-src 'strict-dynamic' 'report-sample'
'nonce-f6393661395af25044660c187c2cd6c4';
report-uri https://ourserver; ,
```

```
script-src 'report-sample'
'nonce-f6393661395af25044660c187c2cd6c8';
report-uri https://ourserver; ,
```

```
object-src 'none' 'report-sample';
report-uri https://ourserver; ,
```

```
base-uri 'none' 'report-sample';
report-uri https://ourserver;
```

CSPs 1, 3, 4 compose the Google's CSP solution at level 3 in Table 1. The CSP violations triggered under them can be used for analyzing if a webpage can adopt a CSP solution at levels 1, 2, or 3.

CSPs 2, 3, 4 compose the Google's CSP solution at level 4 in Table 1. The CSP violations triggered under them can be used for analyzing if a webpage can adopt a CSP solution at level 4.

Fig. 2: Four Inserted CSPs for Webpages.

To achieve a more comprehensive collection of CSP violations by accessing more web resources, the user interaction simulator simulates two common user interactions after each webpage is fully loaded: scrolling down the webpage to the bottom and clicking on the first identified button (which can be a submit, reset, checkbox, or radio button) on the webpage. If the user interaction simulator cannot identify a button to click on the webpage, it will find a link, an input box, or an image to click instead. The request and response collector intercepts all HTTP(S) requests and the corresponding responses generated for a webpage. The HTML document collector saves the loaded HTML document when a webpage is fully loaded. All collected data will be saved in a local database. We consider that a webpage is successfully visited if its HTML document is successfully saved. CSP violations will be extracted from HTTP(S) requests, and original CSPs (if they exist) will be extracted from HTTP(S) response headers. Note that our crawling tool does not remove the original CSPs (if they exist) on webpages when it inserts specified CSPs. Therefore, CSP violations triggered under our inserted CSPs are based on currently allowed resources. Our crawling tool does not check whether there is a CSP deployed via an HTML <meta> tag in a webpage because in practice only a very small number of websites deployed a CSP via an HTML <meta> tag [24].

B. Data Collection and Dataset

We visited webpages of the top 10K websites from the Tranco [18] list (dated November 8th, 2023), using ten computers with identical configurations, from November 2023 to January 2024. Each computer is installed with a Google Chrome browser (Version 104) on a Ubuntu 22.04 LTS system. To analyze the stability of adopting the nonce-based CSP solutions on websites, we performed the website crawling three times. In total, 7,663 websites with 510,056 webpages, 7,603 websites with 504,400 webpages, and 8,162 websites with 558,223 webpages were successfully visited in the first, second, and third crawlings, respectively. As in studies such as [4], we denote a website as an eTLD+1 (effective top-level domain + 1) entity, like google.com or bbc.co.uk. Webpages sharing the same eTLD+1 are considered to be from the same website. We further filtered out duplicate websites and webpages in each crawling, which leaves us 7,034 unique websites with 484,946 webpages, 6,985 unique websites with 470,262 webpages, and 7,427 unique websites with 513,726 webpages in the first, second, and third crawlings, respectively. The analysis of the adoptability of Google's nonce-based CSP solution on websites (Section IV-A) and the adoption issues encountered by websites that would fail to adopt them (Section IV-B) is based on the 7,034 websites visited in the first crawling. The analysis of the stability of the nonce-based CSP solution adoption on websites is based on websites visited in three crawlings (Section IV-C).

C. Adoption Feasibility Evaluation Methods

The feasibility of adopting the nonce-based CSP solutions on a website is evaluated based on the adoptability of nonce-based CSP solutions on each webpage visited in our study. If all visited webpages of a website can adopt the nonce-based CSP solutions at or above level 1, we consider the website can adopt ***the nonce-based CSP solutions***. If all visited webpages of a website can adopt the nonce-based CSP solutions at or above level 3, we consider that the website can adopt ***the stronger nonce-based CSP solutions***.

1) Highest Security Level of CSP Solutions Adoptable for a Webpage: To evaluate the adoptability of the nonce-based CSP solutions on each webpage, we analyze the highest security level of nonce-based CSP solutions it can adopt by using Algorithm 1. Given CSP violations W_Vio generated under the four inserted CSPs of a webpage, and four empty lists $CSP\_solution\_level\_i$ ($i = 1, 2, 3, 4$) for recording the resolvability of CSP violations under each nonce-based CSP solution, the algorithm will automatically return the highest security level H_CSP_Sol of the nonce-based CSP solutions that the webpage can adopt. A CSP violation is considered ***resolvable*** under a nonce-based CSP solution if the CSP solution allows the violating resource in the CSP violation to be loaded or executed by adding nonces or hash values without the web developers' *code refactoring* (i.e., restructuring existing code while preserving its functionality [32]). We differentiate CSP violations generated by the accesses of third-party resources from those generated by the accesses of first-party resources to accurately determine the resolvability of CSP violations under a nonce-based CSP solution.

In more details, under the nonce-based CSP solution at level 4, CSP violations triggered by the accesses of five types of resources are resolvable by adding nonces: (1) inline scripts, (2) first-party scripts directly requested by the current webpage, (3) first-party scripts requested by any already-allowed scripts, (4) scripts inserted into the current webpage by first-party scripts, and (5) third-party scripts directly requested by the current webpage. Under the nonce-based CSP solution at level 3, besides the above five types of resources, CSP violations from two more types of resources are resolvable due to the added "strict-dynamic" directive value: (6) non-parser-inserted scripts inserted into the current webpage by third-party scripts, and (7) third-party scripts requested by any already-allowed scripts. Under the nonce-based CSP solution at level 2, besides the above seven types of resources, CSP violations from two more types of resources are resolvable due to the added " 'unsafe-eval' " directive value: (8) eval-like functions induced by first-party scripts, and (9) eval-like functions induced by third-party scripts. Under the nonce-based CSP solution at level 1, besides the above nine types of resources, CSP violations from one more type of resources are resolvable due to the added " 'unsafe-hashes' " directive value: (10) event handlers inserted into the current webpage by first-party scripts.

In Algorithm 1, H_CSP_Sol is initialized as 0 in Line 1, which indicates that there are no nonce-based CSP solutions adoptable for the webpage. From Line 2 to Line 16, the algorithm determines whether a CSP violation *csp_vio* is resolvable under each nonce-based CSP solution. If a CSP violation *csp_vio* is resolvable under a nonce-based CSP solution, the corresponding list $CSP\_solution\_level\_i$ (i=1,2,3,4) that records the resolvability of CSP violations will add a "RES" value. Otherwise, the corresponding list will add an "UNRES" value to represent that the CSP violation csp_vio is unresolvable under the nonce-based CSP solution. For the nonce-based CSP solution at level 4, the algorithm determines the resolvability of CSP violations generated under the inserted CSPs 2, 3, and 4 (Figure 2). For the solutions at levels 1, 2, and 3, it determines the resolvability of CSP violations generated under the inserted CSPs 1, 3, and 4 (Figure 2).

From Line 17 to Line 22, the algorithm determines the nonce-based CSP solutions that a webpage can adopt. For each nonce-based CSP solution, the algorithm checks the corresponding list $CSP\_solution\_level\_i$ (i=1,2,3,4). If there is no "UNRES" value in the list, we consider the webpage can adopt the corresponding nonce-based CSP solution. The algorithm records the highest security level H_CSP_Sol of nonce-based CSP solutions that a webpage can adopt and returns H_CSP_Sol.

Algorithm 1 Determining the Highest Security Level of Nonce-Based CSP Solutions a Webpage Can Adopt

Input: W_Vio is a set of CSP violations generated under the four inserted CSPs (CSP 1 to CSP 4 in Figure 2) of a webpage
Input: $CSP\_solution\_level\_i$ ($i = 1, 2, 3, 4$) is an empty list to record the resolvability of CSP violations for a certain CSP solution
Output: the highest security CSP solution H_CSP_Sol that the webpage can adopt

```
 1: $H\_CSP\_Sol \leftarrow 0$;
 2: for each csp_vio in $W\_Vio$ do
 3:     for i= 4 to 1 do
 4:         if i==4 and csp_vio is generated by inserted CSP 1 then
 5:             continue;
 6:         end if
 7:         if i!=4 and csp_vio is generated by inserted CSP 2 then
 8:             continue;
 9:         end if
10:         if csp_vio is resolvable at level i then
11:             APPEND($CSP\_solution\_level\_i$, "RES");
12:         else
13:             APPEND($CSP\_solution\_level\_i$, "UNRES");
14:         end if
15:     end for
16: end for
17: for i= 4 to 1 do
18:     if "UNRES" not in $CSP\_solution\_level\_i$ then
19:         $H\_CSP\_Sol \leftarrow i$;
20:         break;
21:     end if
22: end for
23: return $H\_CSP\_Sol$
```

2) Third-party Resource Identification: As in studies such as [4], [27], we use the notion of an eTLD+1 to distinguish first and third parties. To differentiate third parties more accurately by considering that hosts with different eTLD+1 may actually belong to the same party or organization, we refined our results based on the domain relationships provided by the webXray domain owner list [30] and the domain owner list derived by Steffens et al. in [27]. The "blocked-uri", "violated-directive", and "source-file" fields of CSP violations generated by Google Chrome indicate the types (e.g., event handlers, inline scripts, requested URLs, or eval-like functions) of the reported resources and resource sources.

D. Ethical Considerations and Reproducibility

Our data collection process (Section III-B) does not raise obvious ethical issues. It does not involve human subjects or potentially sensitive data of users. Our constructed crawling tool only visited the public webpages of websites. On each webpage, our tool inserts the specified CSPs, scrolls down to the bottom, and only clicks on one button to collect HTTP(S) requests and responses (Section III-A). Therefore, there is minimal or even no change to the click-through rate of ads, and the traffic burden incurred to each visited website is also minimal. In web measurement studies, reproducibility is a requirement that is crucial to sustainable research [1], [6], [12]. Our work is reproducible because we provide the details of our website crawling and data collection framework, our data collection process and dataset, and our data processing and analysis methods in this section as suggested in studies such as [1], [6], [12]; our constructed crawling tool, datasets, and data analysis code are shared on the Zenodo Repository [26].

IV. Results and Analysis

In this section, we answer our RQ1 (Section IV-A), RQ2 (Section IV-B), and RQ3 (Section IV-C). All results in this section are derived using our analysis methods (Section III-C) based on our collected data (Section III-B).

A. RQ1: Adoptability of Nonce-Based CSP solutions

To answer RQ1, we investigate the adoptability of the nonce-based CSP solutions on each webpage visited in our study across websites (Section IV-A1), and analyze the improvement of deployed whitelisting-based CSPs on websites (Section IV-A2).

1) Adoption Patterns of Nonce-Based CSP Solutions Across Websites: We checked the distribution of the 7,034 websites successfully visited in the first crawling regarding the lowest security level of adoptable nonce-based CSP solutions on their webpages visited in our study. If at least one webpage can only adopt a nonce-based CSP solution at a certain security level i (i=1,2,3,4), and each of the remaining webpages is able to adopt nonce-based CSP solution at or above this level, we refer to this website as having ***the lowest security level of adoptable nonce-based CSP solutions*** at level i. Overall, we found that 2,621 (37.26%) of those 7,034 websites cannot adopt the nonce-based CSP solutions since they have at least one of their webpages unable to adopt nonce-based CSP solutions at or above level 1, while 4,413 (62.74%=9.50%+12.03%+23.33%+17.88%) of those 7,034 websites can adopt the nonce-based CSP solutions since all their webpages visited in our study can adopt nonce-based CSP solutions at or above level 1.

Figure 3 illustrates the distribution of those 7,034 websites regarding the percentages of them having the lowest security levels of adoptable nonce-based CSP solutions at 1, 2, 3, and 4, as well as having at least one of their webpages unable to adopt any nonce-based CSP solutions. Specifically, among those 7,034 websites, 9.50%, 12.03%, 23.33%, and 17.88% of them have their lowest security level of adoptable nonce-based CSP solutions at levels 4, 3, 2, and 1, respectively. It is worth noting that 1,514 (21.53%=9.50%+12.03%) of those 7,034 websites can even adopt the stronger nonce-based CSP solutions since all their webpages visited in our study can adopt nonce-based CSP solutions at or above level 3.

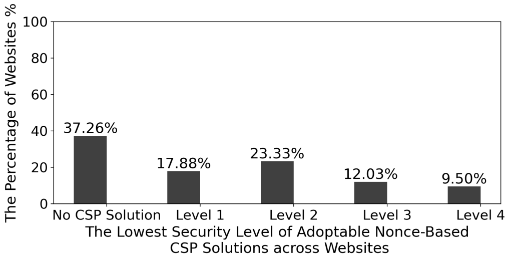


Fig. 3: Distribution of 7,034 websites regarding what percentages of them having the lowest security level of adoptable nonce-based CSP solutions at levels 1, 2, 3, and 4, as well as having at least one of webpages unable to adopt nonce-based CSP solutions.

For each of those 2,621 websites that cannot adopt the nonce-based CSP solutions, we further checked the distribution of its webpages visited in our study regarding what percentages of them can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4, as well as cannot adopt any nonce-based CSP solutions. Figure 4 is the boxplot of such distributions across the 2,621 websites. On average, the percentages of webpages that can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4, and those that cannot adopt any nonce-based CSP solutions, are 10.48%, 25.46%, 10.96%, 13.25%, and 39.85%, respectively, across the 2,621 websites. We can see that for those 2,621 websites, usually the adoption is hard (due to the existence of resources that are causing unresolvable CSP violations) on around 40% of their webpages; meanwhile, a noticeable percentage of webpages can adopt nonce-based CSP solutions at level 2. Additionally, the boxes, the whiskers, and the median values are similar for webpages that can adopt nonce-based CSP solutions at levels 1, 3, and 4.

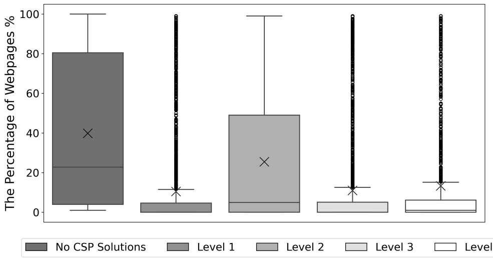


Fig. 4: Distributions of the webpages visited in our study regarding what percentages of them can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4, as well as cannot adopt any nonce-based CSP solutions, respectively, across each of the 2,621 websites that cannot adopt nonce-based CSP solutions. The X marks represent the mean values.

Similarly, for each of those 4,413 websites that can adopt the nonce-based CSP solutions, we further checked its distribution of the webpages visited in our study regarding what percentages of them can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4. Figure 5 shows the boxplot of such distributions across the 4,413 websites. On average, the percentages of webpages that can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4 are 13.53%, 35.08%, 26.33%, and 25.07%, respectively, across the 4,413 websites. The median values of the percentages of webpages that can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4 across the 4,413 websites are 0%, 1.19%, 0% and 1.06%, respectively. We can see that for those 4,413 websites, their webpages can often adopt nonce-based CSP solutions at level 2; meanwhile, many of them can even achieve nonce-based CSP solutions at levels 3 and 4.

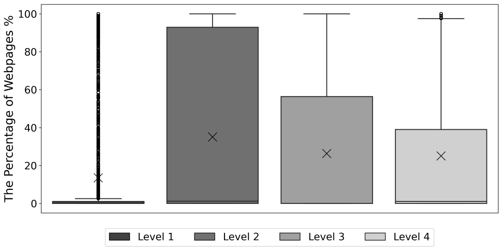


Fig. 5: Distributions of the webpages visited in our study regarding what percentages of them can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4, respectively, across each of the 4,413 websites that can adopt nonce-based CSP solutions. The X marks represent the mean values.

2) Potential CSP Improvement: In total, 113,863 webpages from 3,041 websites deployed CSPs under the enforcement mode. We found that 109,878 (96.50%) out of 113,863 webpages deployed whitelisting-based CSPs. Among these 109,878 webpages, 45,669 (41.45%) deployed whitelisting-based CSPs for content control; 83.46% of these 45,669 webpages deployed whitelisting-based CSPs with an " 'unsafe-inline' " directive value which allows any inline or embedded scripts to be loaded or executed; 54.79% of them deployed whitelisting-based CSPs with a "*" directive value or scheme directive values (i.e., an "http:", "https:", or "data:" directive value) which allow resources from any source URLs to be loaded or executed. We explored the feasibility of adopting the nonce-based CSP solutions on those 109,878 webpages. We found that only 16.95% of them cannot adopt any nonce-based CSP solutions (because they contain at least one of the following four types of resources that can cause adoption issues for any of the four nonce-based CSP solutions: (1) URLs in <base> tags; (2) plugins in <object>, <embed>, or <applet> tags; (3) parser-inserted scripts inserted by third-party scripts; (4) event handlers inserted by third-party scripts); meanwhile, 17.34%, 27.12%, 17.36%, and 21.23% of them can adopt nonce-based CSP solutions at levels 1, 2, 3, and 4, respectively. We can see that most (83.05%) webpages that deployed whitelisting-based CSPs can adopt the nonce-based CSP solutions, and a noticeable number (38.59%) of them can even adopt stronger ones.

In addition, we found that 439 websites deployed whitelisting-based CSPs on all their webpages visited in our study, and 372 of them deployed the same whitelisting-based CSP on all their webpages. Among those 439 websites, 55.12% of them can adopt the nonce-based CSP solutions, and 21.41% of them can even adopt the stronger nonce-based CSP solutions. Among those 372 websites, 152 deployed whitelisting-based CSPs for content control, of which only 19 have no security issues (i.e., they did not use an " 'unsafe-inline' " directive value, a "*" directive value, or scheme directive values); meanwhile, 57.52% of those 372 websites can adopt the nonce-based CSP solutions; 24.19% of them can even adopt the stronger ones.

B. RQ2: Adoption Issues of Nonce-Based CSP Solutions

To answer RQ2, we investigate the issues of adopting the weaker nonce-based CSP solutions on webpages that cannot adopt any nonce-based CSP solutions (Section IV-B1), and the issues of adopting the stronger nonce-based CSP solutions on webpages that can only adopt weaker ones (Section IV-B2). We also explore the promotion of the nonce-based CSP solutions on websites by addressing adoption issues caused by first-party resources (Section IV-B3). We used our analysis methods presented in Section III-C to explore the issues that prevent certain webpages from adopting the nonce-based CSP solution at a specific level and correspondingly how developers may address the issues. In more details, we extracted the information of violating resources (e.g., source files, types, and the violated CSP directives as introduced in Section II-A) from the CSP violations on each of those webpages, and identified whether a violating resource is from the first party or a third party website. We further calculated the percentages of webpages that have different types of adoption issues (based on the triggered CSP violations due to the accesses of the corresponding types of resources) that should be addressed by web developers in order to adopt the nonce-based CSP solution at a specific level.

1) Upgrade to the Weaker Nonce-Based CSP Solutions: Overall, 2,621 websites contain 85,523 webpages that cannot adopt any nonce-based CSP solutions. To adopt nonce-based CSP solutions at level 1 on those 85,523 webpages, 45.78% and 12.89% of those 2,621 websites need to address the adoption issues of URLs fetched for first-party and third-party resources in <base> tags, respectively; 7.44% and 0.69% of them need to address the adoption issues of plugins embedded into <object>, <embed>, or <applet> tags by first-party and third-party scripts, respectively; 54.14% of them need to address the adoption issues of parser-inserted scripts inserted by third-party scripts into webpages; 20.95% of them need to address the adoption issues of event handlers inserted by third-party scripts. This 20.95% result differs from what prior studies opined that third-party event handlers were a major reason for websites to deploy CSPs with an " 'unsafe-inline' " value instead of using nonces [25], [27]. To adopt nonce-based CSP solutions at level 2 on webpages that cannot adopt any nonce-based CSP solutions, besides addressing the issues of adopting nonce-based CSP solutions at level 1, 42.54% of those 2,621 websites need to further address the adoption issues of event handlers inserted by first-party scripts.

2) Upgrade to the Stronger Nonce-Based CSP Solutions: Overall, 2,389 websites contain 67,878 webpages for which the highest security level of adoptable nonce-based CSP solutions is level 1. To adopt nonce-based CSP solutions at level 3 on those 67,878 webpages, all and none of these 2,389 websites need to address the adoption issues of event handlers inserted by first-party and third-party scripts, respectively; 43.11% and 53.99% of those 2,389 websites need to address the adoption issues of eval-like functions induced by first-party and third-party scripts, respectively. To adopt nonce-based CSP solutions at level 4 on those 67,878 webpages, besides addressing the adoption issues of eval-like functions and event handlers as adopting nonce-based CSP solutions at level 3, 84.55% of those 2,389 websites need to further address the adoption issues of third-party scripts requested by other scripts; 48.68% of them need to further address the adoption issues of scripts inserted by third-party scripts.

Overall, 4,139 websites contain 163,568 webpages for which the highest security level of adoptable nonce-based CSP solutions is level 2. To adopt nonce-based CSP solutions at level 3 on these 163,568 webpages, 68.40% and 74.80% of those 4,139 websites need to address the adoption issues of eval-like functions induced by first-party and third-party scripts, respectively. To adopt nonce-based CSP solutions at level 4 on those 163,568 webpages, besides addressing the adoption issues of eval-like functions as adopting the nonce-based CSP solutions at level 3, 88.11% of those 4,139 websites need to further address the adoption issues of third-party scripts that are requested by other scripts; 56.15% of them need to further address the adoption issues of scripts that are inserted by third-party scripts.

3) Promotion of the Nonce-Based CSP Solutions by Addressing Adoption Issues Caused by First-Party Resources: The adoption issues of the nonce-based CSP solutions caused by third-party resources may be hard to be addressed as the cooperation from the third parties would be needed; however, web developers can possibly address the adoption issues caused by first-party resources. Figure 6 illustrates the potential changes of the lowest security level of adoptable nonce-based CSP solutions across 7,034 websites if adoption issues caused by first-party resources would be addressed. We found that if websites can address the adoption issues caused by first-party resources, the total number of websites that can adopt the nonce-based CSP solutions will increase by 13.39%. The number of websites having the lowest security level of adoptable nonce-based CSP solutions at levels 2, 3, and 4 will increase by 8.64%, 15.74%, and 6.90%, respectively; the number of websites that can adopt the stronger nonce-based CSP solutions will increase by 22.64%. All websites having the lowest security level of adoptable nonce-based CSP solutions at level 1 will have their lowest security level at level 2 since all event handlers inserted into their webpages are by first-party resources.

In more details, among the 2,621 websites that contain webpages unable to adopt any nonce-based CSP solutions, none of them would have the lowest security level of adoptable nonce-based CSP solutions at level 1, while 17.17%, 9.31%, and 9.46% of them would have the lowest security level of adoptable nonce-based CSP solutions at levels 2, 3, and 4, respectively. Among the 1,258 websites that have the lowest security level of adoptable nonce-based CSP solutions at level 1, 55.24%, 36.64%, and 8.12% of them would have the lowest security level of adoptable nonce-based CSP solutions at levels 2, 3 and 4, respectively. Among the 1,641 websites that have the lowest security level of adoptable nonce-based CSP solutions at level 2, 24.50% and 8.23% of them would have the lowest security level of adoptable nonce-based CSP solutions at levels 3 and 4, respectively. There are no promotions for websites that can adopt the stronger nonce-based CSP solutions since their webpages do not contain first-party resources that can cause adoption issues.

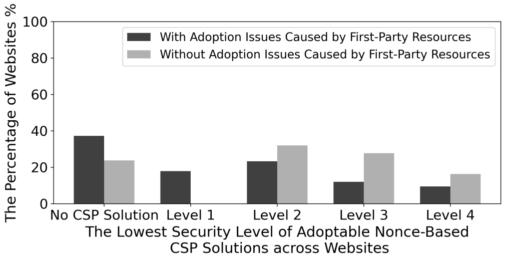


Fig. 6: Changes of the lowest security level of adoptable nonce-based CSP solutions across 7,034 websites if adoption issues caused by first-party resources would be addressed.

C. RQ3: Adoption Stability Across Crawlings

To answer RQ3, we compare the results across the three crawlings. Overall, we found that the results from the three crawlings are similar. The percentages of websites that can adopt the nonce-based CSP solutions in the first, second, and third crawlings are 62.74%, 63.71%, and 63.16%, respectively. The percentages of websites that can adopt the stronger nonce-based CSP solutions in the first, second, and third crawlings are 21.53%, 23.41%, and 22.12%, respectively. Meanwhile, 5,797 websites were commonly visited in the three crawlings, among which 59.56% can always adopt the nonce-based CSP solutions, while 34.91% always cannot adopt the nonce-based CSP solutions. Additionally, 19.32% of those 5,797 websites can always adopt the stronger nonce-based CSP solutions. We analyze the causes of variations across crawlings as follows.

1) Variations Caused by Newly Appeared and Disappeared Websites: Newly appeared websites and disappeared websites can cause variations in the results of the nonce-based CSP solution adoption. We found that in the second crawling, 954 websites that were successfully visited in the first crawling disappeared, and 815 websites newly appeared. Among the 954 websites that disappeared in the second crawling, 620 and 227 of them can adopt the nonce-based CSP solutions and the stronger ones, respectively. Among the 815 websites that newly appeared in the second crawling, 586 and 266 of them can adopt the nonce-based CSP solutions and the stronger ones, respectively. Overall, in the second crawling, newly appeared websites and disappeared websites together make the number of websites that can adopt the nonce-based CSP solutions and the stronger ones decrease by 34 and increase by 39, respectively, compared to the first crawling.

In the third crawling, 509 websites that were successfully visited in the second crawling disappeared, and 1,041 websites newly appeared. Among the 509 websites that disappeared in the third crawling, 370 and 169 of them can adopt the nonce-based CSP solutions and the stronger ones, respectively. Among the 1,041 websites that newly appeared in the third crawling, 675 and 238 of them can adopt the nonce-based CSP solutions and the stronger ones, respectively. Overall, in the third crawling, newly appeared websites and disappeared websites together make the number of websites that can adopt the nonce-based CSP solutions and the stronger ones increase by 305 and 69, respectively, compared to the second crawling.

2) Variations Caused by Newly Appeared and Disappeared Webpages: Websites commonly visited but with different webpages in two crawlings can cause variations in the adoption results. Specifically, if webpages that cannot adopt any nonce-based solutions (or the stronger ones) newly appear, the website can no longer adopt the nonce-based CSP solutions (or the stronger ones) because not all our visited webpages of it can adopt the nonce-based CSP solutions (or the stronger ones). Conversely, if all such webpages disappear, the website can adopt the nonce-based CSP solutions (or the stronger ones) because all our visited webpages of it can adopt the nonce-based CSP solutions (or the stronger ones).

We found that 4,470 websites that were commonly visited in the first and second crawlings have different webpages, among which 4,326 have webpages that were successfully visited in the first crawling but disappeared in the second crawling, and 4,301 have new webpages in the second crawling. Among those 4,326 websites, 49.01% of them have less than 20% webpages disappeared; among those 4,301 websites, 49.77% of them have less than 20% webpages newly appeared. Among those 4,326 websites, 92 and 110 become able to adopt the nonce-based CSP solutions and the stronger ones, respectively, due to their disappeared webpages. Among those 4,301 websites, 78 and 46 can no longer adopt the nonce-based CSP solutions and the stronger ones, respectively, due to their newly appeared webpages. Overall, in the second crawling, newly appeared webpages and disappeared webpages together make the number of websites that can adopt the nonce-based CSP solutions and the stronger ones increase by 14 and 64, respectively, compared to the first crawling.

In the second and third crawlings, 4,661 websites were commonly visited but have different webpages. Among those 4,661 websites, 4,474 have webpages disappeared in the third crawling, and 4,535 have newly appeared webpages in the third crawling. Among those 4,474 websites, 48.16% of them have less than 20% webpages disappeared; among those 4,535 websites, 47.56% of them have less than 20% webpages newly appeared. Among those 4,474 websites, 98 and 67 become able to adopt the nonce-based CSP solutions and the stronger ones, respectively, due to their disappeared webpages. Among those 4,535 websites, 112 and 117 can no longer adopt the nonce-based CSP solutions and the stronger ones, respectively, due to their newly appeared webpages. Overall, in the third crawling, newly appeared webpages and disappeared webpages together make the number of websites that can adopt the nonce-based CSP solutions and the stronger ones decrease by 14 and 50, respectively, compared to the second crawling.

3) Variations Caused by Webpages with Changes on the Types of the Resources Used: A webpage may change its content in different crawlings. If some webpages can no longer adopt the nonce-based CSP solutions (or the stronger ones) due to changes in their resource types (e.g., using event handlers inserted by third-party scripts in the next crawling), the website can no longer adopt the nonce-based CSP solutions (or the stronger ones) because not all our visited webpages of it can adopt the nonce-based CSP solutions (or the stronger ones). Conversely, if all webpages become able to adopt the nonce-based CSP solutions (or the stronger ones) due to changes in their resource types (e.g., no URLs in <base> tags in the next crawling), the website can adopt the nonce-based CSP solutions (or the stronger ones).

We found that 2,168 websites that were commonly visited in the first and second crawlings contain webpages with resource type changes, and 74.35% of them have less than 20% webpages with resource type changes. Among those 2,168 websites, 36 can no longer adopt the nonce-based CSP solutions in the second crawling due to webpage resource type changes, while 38 become able to adopt the nonce-based CSP solutions due to webpage resource type changes; 35 can no longer adopt the stronger nonce-based CSP solutions in the second crawling due to webpage resource type changes, while 29 become able to adopt the stronger nonce-based CSP solutions due to webpage resource type changes. Overall, in the second crawling, resource type changes make the number of websites that can adopt nonce-based CSP solutions and the stronger ones increase by two and decrease by six, respectively, compared to the first crawling.

In the second and third crawlings, 2,084 websites that were commonly visited contain webpages with resource type changes, and 73.41% of them have less than 20% webpages with resource type changes. Among those 2,084 websites, 40 can no longer adopt the nonce-based CSP solutions in the third crawling due to webpage resource type changes, while 51 become able to adopt the nonce-based CSP solutions due to webpage resource type changes; 26 can no longer adopt the stronger nonce-based CSP solutions in the third crawling due to webpage resource type changes, while 43 become able to adopt the stronger nonce-based CSP solutions due to webpage resource type changes. Overall, in the third crawling, resource type changes make the number of websites that can adopt the nonce-based CSP solutions and the stronger ones increase by 11 and 17, respectively, compared to the second crawling.

In addition, in the second crawling, 20 and 15 websites can no longer adopt the nonce-based CSP solutions and the stronger ones, respectively, due to newly appeared webpages and resource type changes on some other webpages; 22 and 12 websites become able to adopt the nonce-based CSP solutions and the stronger ones, respectively, due to disappeared webpages and resource type changes on some other webpages. In the third crawling, 25 and 14 websites can no longer adopt the nonce-based CSP solutions and the stronger ones, respectively, due to newly appeared webpages and resource type changes on some other webpages; 29 and 21 websites become able to adopt the nonce-based CSP solutions and the stronger ones, respectively, due to disappeared webpages and resource type changes on some other webpages.

V. Discussion

Summary of Four Key Findings. First, most websites can adopt the nonce-based CSP solutions, and a noticeable percentage of them can even adopt the stronger ones (Section IV-A1); this is a very encouraging message that should be conveyed to the developers or administrators of those websites, for them to actively learn and adopt nonce-based CSP solutions. For websites that cannot adopt the nonce-based CSP solutions, usually the adoption is hard on around 40% of their webpages; since CSP is a page-level protection mechanism, it is still wise for those websites to deploy nonce-based CSP solutions on all other possible webpages following the principle of least privilege. Second, most webpages that already deployed whitelisting-based CSPs can adopt nonce-based CSP solutions which are simpler and more secure (Section IV-A2). Meanwhile, most websites that deployed the same whitelisting-based CSP on all their webpages visited in our study can adopt the nonce-based CSP solutions. Third, both first-party resources and third-party resources can cause issues of adopting the nonce-based CSP solutions on websites (Sections IV-B1 and IV-B2). If websites move forward to address the adoption issues caused by their first-party resources, there will be noticeable increases in the number of websites that can adopt the nonce-based CSP solutions and even the stronger ones (Section IV-B3). Fourth, the percentages of websites that can adopt the nonce-based CSP solutions are similar across the three crawlings (Section IV-C). The appearance and disappearance of websites and webpages, as well as resource type changes on webpages, have a minor impact on the adoption stability. These findings are new and were not reported in existing studies that we reviewed in Section II-B; they expand upon the existing knowledge on promoting the proper deployment of CSPs specifically along the direction of adopting Google's nonce-based CSP solutions on websites.

Recommendations. Based on the key findings from our study, we provide three recommendations that may help developers adopt Google's nonce-based CSP solutions. First, for developers who deployed whitelisting-based CSPs using a "*" or an " 'unsafe-inline' " directive value, we recommend them to check the feasibility of adopting the nonce-based CSP solutions on their webpages (e.g., by using our crawling tool [26] to visit their website and analyzing the CSP violation report, or by manually deploying a solution shown in Table I under the report-only mode and analyzing the CSP violations via the DevTools console of a browser [10], [14]). Second, for developers who are adopting nonce-based CSP solutions but encountering issues (e.g., due to content changes as described in Section IV-C3) on their webpages, we recommend them to either avoid using or refactor their first-party resources (e.g., eval-like functions, parser-inserted scripts, and event handlers [32]) that can cause the adoption issues. Meanwhile, we recommend them to use third-party resources only if those resources will not cause an issue of adopting nonce-based CSP solutions, unless using them is absolutely necessary. Finally, for developers of third-party resources, we recommend them to avoid providing scripts (e.g., scripts that insert parser-inserted scripts into webpages, embed event handlers into webpages, or use eval-like functions) that can cause adoption issues of the nonce-based CSP solutions on webpages. We also recommend them to explicitly describe the behaviors of their provided scripts to facilitate the proper CSP deployment on webpages.

Limitations and Future Work. The major threat to the validity of our study comes from the dynamic analysis (which is necessary due to the dynamic web content generation from both the server and the client sides) nature of our methodology. Especially, due to the ethical (Section III-D) and effort considerations, we only visited 101 webpages of each website in the Tranco top 10K list, and stayed up to 60 seconds on a webpage with two simulated user interactions. These may not be sufficient to represent an entire visited website in terms of the internal validity, and our findings may not be generalized to unvisited websites in terms of the external validity. Future studies may intensively visit more webpages on more websites (in both stateless and stateful manners) to check whether other adoption issues also exist. Our study is also limited in that we did not solicit feedback from web developers on our findings (which also raises internal validity concerns) and did not further investigate the ways to persuade them to take the actual adoption actions (which raises ecological validity concerns). Interviewing web developers and exploring the ways that can effectively inform them about the feasibility and support them with the details (such as the detailed adoption issues on each website and each webpage as we analyzed) can be valuable in the future. In addition, while our RQ3 results in Section IV-C indicate good adoption stability across crawlings, we did not zoom into the analysis of the impact of different website structures (e.g., single-page, multi-page, dynamic content-heavy, and CDN-varying applications) on the adoption rates. Nonce-based CSP solutions provide a way to flexibly allow dynamic and third-party resources (Section II-A); thus, we speculate that website structure factors' impact on the deployment of nonce-based CSPs is limited, but this should be verified in future studies.

VI. Conclusion

In this paper, we analyzed the feasibility of adopting Google's nonce-based CSP solutions on websites. We found that among 7,034 websites, 62.74% of them can adopt the nonce-based CSP solutions, and 21.53% of them can even adopt the stronger ones. For websites that cannot adopt the nonce-based CSP solutions, usually the adoption is hard on around 40% of their webpages. Meanwhile, if developers move forward to address the adoption issues caused by their first-party resources, the number of websites that can adopt nonce-based CSP solutions and even the stronger ones will increase noticeably. Additionally, the results of the feasibility of adopting the nonce-based CSP solutions on websites are similar across the three crawlings. These results are very encouraging, and we further provided recommendations to help websites adopt nonce-based CSP solutions and improve their deployed CSPs.

REFERENCES

[1] Syed Suleman Ahmad, Muhammad Daniyal Dar, Muhammad Fareed Zaffar, Narseo Vallina-Rodriguez, and Rishab Nithyanand. Apophanies or epiphanies? how crawlers impact our understanding of the web. In *Proceedings of the ACM Web Conference (WWW)*, 2020.

[2] Stefano Calzavara, Alvise Rabitti, and Michele Bugliesi. Content Security Problems?: Evaluating the Effectiveness of Content Security Policy in the Wild. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[3] Stefano Calzavara, Alvise Rabitti, and Michele Bugliesi. Semantics-Based Analysis of Content Security Policy Deployment. *ACM Transactions on the Web (TWEB)*, 12(2), 2018.

[4] Stefano Calzavara, Tobias Urban, Dennis Tatang, Marius Steffens, and Ben Stock. Reining in the Web's Inconsistencies with Site Policy. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[5] CWE Top 25 Most Dangerous Software Weaknesses, 2024. https://cwe.mitre.org/top25/.

[6] Nurullah Demir, Matteo Große-Kampmann, Tobias Urban, Christian Wressnegger, Thorsten Holz, and Norbert Pohlmann. Reproducibility and replicability of web measurement studies. In *Proceedings of the ACM Web Conference (WWW)*, 2022.

[7] Mattia Fazzini, Prateek Saxena, and Alessandro Orso. AutoCSP: Automatically Retrofitting CSP to Web Applications. In *Proceedings of the IEEE/ACM IEEE International Conference on Software Engineering (ICSE)*, 2015.

[8] Matteo Golinelli, Francesco Bonomi, and Bruno Crispo. The Nonce-nce of Web Security: An Investigation of CSP Nonces Reuse. In *Proceedings of the European Symposium on Research in Computer Security (ESORICS) International Workshops*, 2023.

[9] Content Security Policy — Privacy & Security — Chrome for Developers, 2024. https://developer.chrome.com/docs/privacy-security/csp.

[10] Google Chrome DevTools Console, 2024. https://developer.chrome.com/docs/devtools/console.

[11] Florian Hantke, Stefano Calzavara, Moritz Wilhelm, Alvise Rabitti, and Ben Stock. You call this archaeology? evaluating web archives for reproducible web security measurements. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2023.

[12] Jordan Jueckstock, Shaown Sarker, Peter Snyder, Aidan Beggs, Panagiotis Papadopoulos, Matteo Varvello, Benjamin Livshits, and Alexandros Kapravelos. Towards realistic and reproducible web crawl measurements. In *Proceedings of the ACM Web Conference (WWW)*, 2021.

[13] Content Security Policy (CSP) - HTTP — MDN, 2024. https://developer.mozilla.org/en-US/docs/Web/HTTP/CSP.

[14] Mozilla Firefox DevTools Console, 2024. https://firefox-source-docs.mozilla.org/devtools-user/index.html.

[15] OWASP Top Ten — OWASP Foundation, 2024. https://owasp.org/www-project-top-ten/.

[16] Cross Site Scripting (XSS) — OWASP Foundation, 2024. https://owasp.org/www-community/attacks/xss/.

[17] Xiang Pan, Yinzhi Cao, Shuangping Liu, Yu Zhou, Yan Chen, and Tingzhe Zhou. CSPAutoGen: Black-box Enforcement of Content Security Policy upon Real-world Websites. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[18] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. 2019.

[19] Puppeteer, 2024. https://pptr.dev/.

[20] Mengxia Ren and Chuan Yue. Coverage and Secure Use Analysis of Content Security Policies via Clustering. In *Proceedings of the IEEE European Symposium on Security and Privacy (EuroSP)*, 2023.

[21] Mengxia Ren and Chuan Yue. Content Security Policy Deployment Issues Related to Third-party Scripts among Builder-generated Websites and Other Websites. In *Proceedings of the IEEE International Performance, Computing, and Communications Conference (IPCCC)*, 2024.

[22] Part of Hypertext Transfer Protocol – HTTP/1.1 RFC 2616, 2024. https://www.w3.org/Protocols/rfc2616/rfc2616-sec4.html#sec4.2.

[23] RFC 9110 HTTP Semantics, 2024. https://www.rfc-editor.org/rfc/rfc9110.html#name-field-lines-and-combined-fi.

[24] Sebastian Roth, Timothy Barron, Stefano Calzavara, Nick Nikiforakis, and Ben Stock. Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2020.

[25] Sebastian Roth, Lea Gröber, Michael Backes, Katharina Krombholz, and Ben Stock. 12 Angry Developers - A Qualitative Study on Developers' Struggles with CSP. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2021.

[26] The Crawling Tool, Datasets, and Data Analysis Code of This Paper Shared through the EU Zenodo Open Science Repository (please check its latest version), January 2025. https://zenodo.org/records/14751981.

[27] Marius Steffens, Marius Musch, Martin Johns, and Ben Stock. Who's hosting the block party? studying third-party blockage of csp and sri. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, 2021.

[28] W3C: Content Security Policy Level 3, 2024. https://www.w3.org/TR/CSP3/.

[29] W3C: Content Security Policy Level 3: The effect of multiple policies, 2024. https://www.w3.org/TR/CSP3/#multiple-policies.

[30] webXray Domain Owner List, 2024. https://web.archive.org/web/20200604213008/https://github.com/timlib/webXray_Domain_Owner_List/blob/master/domain_owners.json.

[31] Lukas Weichselbaum. Mitigate cross-site scripting (XSS) with a strict Content Security Policy. 2024. https://web.dev/articles/strict-csp.

[32] Lukas Weichselbaum and Michele Spagnuolo. Content Security Policy - A Successful Mess between Hardening and Mitigation. 2024. https://tinyurl.com/yyohn6o6.

[33] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. CSP Is Dead, Long Live CSP! On the Insecurity of Whitelists and the Future of Content Security Policy. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[34] Michael Weissbacher, Tobias Lauinger, and William K. Robertson. Why Is CSP Failing? Trends and Challenges in CSP Adoption. In *Proceedings of the International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, 2014.