IEICE TRANS. INF. & SYST., VOL.E104–D, NO.11 NOVEMBER 2021

PAPER Special Section on Next-generation Security Applications and Practice

The Uncontrolled Web: Measuring Security Governance on the Web

Yuta TAKATA[†a)], ***Member***, **Hiroshi KUMAGAI**[†], ***and*** **Masaki KAMIZONO**[†], ***Nonmembers***

SUMMARY While websites are becoming more and more complex daily, the difficulty of managing them is also increasing. It is important to conduct regular maintenance against these complex websites to strengthen their security and improve their cyber resilience. However, misconfigurations and vulnerabilities are still being discovered on some pages of websites and cyberattacks against them are never-ending. In this paper, we take the novel approach of applying the concept of security governance to websites; and, as part of this, measuring the consistency of software settings and versions used on these websites. More precisely, we analyze multiple web pages with the same domain name and identify differences in the security settings of HTTP headers and versions of software among them. After analyzing over 8,000 websites of popular global organizations, our measurement results show that over half of the tested websites exhibit differences. For example, we found websites running on a web server whose version changes depending on access and using a JavaScript library with different versions across over half of the tested pages. We identify the cause of such governance failures and propose improvement plans.

key words: *security governance, web measurement, cyber resilience*

1. Introduction

A website is one of the representative systems of a company and provides various services to many users. Hence, if these websites are compromised or taken down by cyberattacks, the economic damage can be significant. In addition to the economic damage, there is also a risk of successful attacks being leveraged by attackers to launch further attacks. A security vendor reported that attackers distributed malicious code and ransomware via dozens of compromised websites [1]. To protect websites from such cyberattacks, various guidelines regarding website management and operations have been published [2], [3]. The guidelines describe basic web security, such as security-related settings, software updating, and vulnerability checks in order to support the operation of secure websites. However, misconfigurations and vulnerabilities are still being discovered on websites, and cyberattacks against them are never-ending [4].

As the size of companies and organizations grows, so does the number of services on their websites. These websites often rely heavily on third-party content to provide their services [5], [6]. For instance, they may make use of web advertisements, analytics, social media, and externally hosted JavaScript libraries to develop features quickly. However, as a result, websites become both more dependent on third-parties' components and more complex. The complexity leads to weaker web security on several pages and might increase the attack surface of the websites [7], [8]. For example, Lauinger et al. researched websites with inclusions of multiple different jQuery versions in the same document. They reported that a part of these duplicate inclusions was caused by dynamic inclusions of libraries by third-party components which can lead to potentially nondeterministic behaviour with respect to vulnerabilities [9]. Of course, it is important to address these individual security issues, but it is also important to apply them to all web pages consistently. Since different software versions have different vulnerabilities, using various versions leads to increase the risk of vulnerability abuse. To minimize the risk, it is desirable to control and manage which versions of which software are used on all web pages with the consistency. In addition, we can expect to accelerate new vulnerability responses by the consistency, as a result, improve their cyber resilience. Therefore, we should strive to apply consistent and governed security rules across all pages of complex websites. The importance of maintaining a certain level of security by applying governance is also described in various standards such as NIST SP800 [2] and ISO/IEC 27014 [10], and complying with them will lead to drive stable business. Although many researchers have studied the state of web technologies, settings, dependencies, and security on individual web pages, they did not focus on multiple pages and provide a comprehensive analysis across them [5]–[9].

In this paper, we take the novel approach of applying the concept of security governance to websites; as part of this, measuring the consistency of software settings and versions used on these websites. Our main motivation is to investigate whether website maintenance and security measures are pervasive when applied across all pages of a website. To understand the *web security governance* of a given website, we crawl multiple web pages of the same domain name and identify differences in the security settings of HTTP headers and versions of software among them. Our simple approach of analyzing multiple web pages can help us find misconfigurations and vulnerabilities newly. We also analyze the details of the identified differences to infer their causes and improve web security. Overall, our study makes the following contributions:

- We shed light on web security governance as it relates to the consistency of website maintenance and security

Manuscript received January 27, 2021.
Manuscript revised May 26, 2021.
Manuscript publicized July 8, 2021.

[†]The authors are with Deloitte Tohmatsu Cyber LLC, Tokyo, 100–0005 Japan.
a) E-mail: yuta.takata@tohmatsu.co.jp

DOI: 10.1587/transinf.2021NGP0003

measures.
- We analyze 8,190 websites, including 77,599 pages of famous global organizations, and show that 4,680 (57.1%) of them exhibit differences concerning HTTP headers and/or software versions between web pages.
- We show that there are websites running on Nginx whose version changes depending on access and websites using different jQuery versions on different pages of the same domain name. Using various versions leads a higher risk of containing vulnerabilities.
- Based on our measurement results, we contribute to accelerating the adoption of security governance and the improvements of cyber resilience on the web.

2. Background

2.1 Web Security

Various guidelines describe web security measures for managing and operating websites more securely [3]. In these guidelines, regular security checks are recommended for web servers (e.g., OS, server software, and middleware) and web applications (e.g., JavaScript libraries, frameworks, and CMS (Content Management Systems)). It is important to check technologies used on websites, version updating, and software patching to prevent abuses of known vulnerabilities in these servers and applications.

In addition to the above web security measures, we can activate the security features of browsers through HTTP response headers [11]. The `Content-Security-Policy`, `Strict-Transport-Security`, and `X-XSS-Protection` headers, shown in Table 1 are examples of such headers. By setting these headers (hereinafter, they are called "HTTP security headers"), we can force browsers to use only HTTPS connections and/or prevent injection attacks although some of them, i.e., XFO and XSSP, are no longer supported in modern browsers. Still, more effective web security can be applied to websites by using both server-side security features and client-side security features.

2.2 Web Security Governance

To maximize the effectiveness of web security measures as described in the previous section, it is important to define and apply a standard, methodology, and process for the management of websites in an organization. By borrowing the concept of security governance, which involves risk management, awareness, reporting, and accountability related to information security in an organization, and applying it to websites, we can expect to reduce inadequate web security. For instance, in a website that has multiple pages, common software settings and versions should be used across all of these pages. Since different versions of web technologies have different vulnerabilities, using various versions leads to increase the risk of vulnerability abuse. To minimize the risk, it is desirable to control and manage which versions of which technologies are used on all web pages with the consistency. In addition to reducing inadequate web security, we can expect to accelerate new countermeasures and new vulnerability responses by applying governance, as a result, improve their cyber resilience. However, misconfigurations and vulnerabilities are still being discovered on websites. We can assume that this is because companies experienced governance failures that in turn, had negative impacts on web security. As an example case, a security vendor reported attackers launching a malware campaign that injected fake JavaScript libraries which imitated legitimate libraries [12]. We speculate that the campaign targeted websites without governance and that the fake library injections could have been discovered early if they were controlled and managed correctly.

In this paper, we investigate the consistency of software settings and versions used on websites as part of security governance. We identify causes of governance failures and consider improvement plans. Although there are many studies that analyzed web security on a page of a website, to the best of our knowledge, there are no studies that measure the governance of web security that can be revealed by analyzing multiple different pages.

3. Related Work

3.1 Web Measurement

There are numerous measurement studies aiming to investigate web security features and their impact [8], [13]–[15].

Table 1 HTTP response headers that enable client security features

Header Name	Description
`Content-Security-Policy` (CSP)	This header is a security policy mechanism that mitigates a wide range of data injections vulnerabilities, such as cross-site scripting (XSS). The mechanism allows browsers to load resources based on a specified policy.
`Strict-Transport-Security` (HSTS)	This header is a security policy mechanism that can force browsers to use HTTPS instead of HTTP for website connections. It specifies its own expiration time and whether the scope includes its subdomains.
`X-Content-Type-Options` (XCTO)	This header forces browsers to load content with a MIME type advertised in a `Content-Type` header. It can be used to mitigate MIME type sniffing.
`X-Frame-Options` (XFO)	This header indicates whether a browser should be allowed to render pages within frames to mitigate clickjacking attacks. However, it includes a feature for legacy browsers, i.e., `ALLOW-FROM`.
`X-XSS-Protection` (XSSP)	This header is a security feature of browsers that can stop pages from rendering when they detect XSS attacks. However, it is for legacy browsers, and the above `Content-Security-Policy` header is recommended to detect unsafe-inline scripts instead in modern browsers.

Van Goethem et al. performed a large-scale security assessment of websites in the EU [13]. They studied the severity of certain vulnerabilities and web security failures, and showed a relationship between both countries and website popularity. Tajalizadehkhoob et al. conducted an empirical analysis of the distribution of web security features and software patching practices in shared hosting providers [14]. They disentangled the defensive efforts of providers and webmasters, and assessed their impact on web compromises. Stock et al. investigated changes in client-side technologies such as HTTP security headers and JavaScript libraries using internet archives [8]. Many researchers also investigated web dependencies of third-party resources and their impact on security [6], [7]. They reported that complicated dependencies prevented HTTPS adoption and led to the implicit loading of malicious content. Mozilla also reported scan results of the Alexa Top 1M websites using the Mozilla Observatory [15]. The report surveyed the pervasiveness of HTTP security headers on a wide range of websites.

The above studies measured the state of web technologies, settings, dependencies, and security on individual web pages. However, the objective of our study is to analyze the consistency, commonality, and governance among multiple web pages.

3.2 Analysis of HTTP Security Headers

HTTP security headers can add security features to websites by simply setting them on the web servers. However, security researchers have found inconsistencies in how these headers are handled among modern browsers, e.g., Microsoft Edge, Mozilla Firefox, and Google Chrome. For example, there are many studies focusing on `Content-Security-Policy` and `X-Frame-Options` headers, which report details of certain discrepancies and their security impact [16]–[18]. Other researchers also found vulnerabilities and bypass methods of features in the `Content-Security-Policy` header [19], [20]. Our study also confirmed some results associated with these studies and complemented their findings, i.e., multiple HTTP headers (see Sect. 5.2.2).

3.3 Analysis of Web Technologies

There are many studies that analyze vulnerabilities and misconfigurations of various web technologies. Vasek et al. conducted a case-control study to identify risk factors that are associated with higher and lower rates of web server compromise [21]. Other researchers also investigated third-party JavaScript libraries and their security impact, known vulnerabilities and the adoption of patches [5], [9]. However, these studies analyzed only one page per website and did not draw conclusions across multiple pages.

3.4 Multi-page Analysis for Security and Privacy

There are numerous studies on characterizing website structures and graphs by analyzing multiple web page over two decades [22]. Several researchers tried to utilize them for web security and privacy purposes [23], [24]. Soska et al. proposed an approach for predicting website compromise using machine learning [23]. They utilized website structures collected by crawling internal links as one of the feature vectors. Urban et al. conducted a large-scale measurement study of the Web dynamics to gain insights into the usage of libraries provided by third-parties [24]. They showed that more information can be collected by crawling websites more deeply. Both of the above studies employed a multi-page analysis similar to ours, but the objectives and targets of each measurement were different.

4. Measuring Web Security Governance

We measure web security governance by analyzing differences in the security of multiple pages in a website. We show our measurement process in Fig. 1. First, we collect and crawl URLs of landing pages. Next, we use the data gathered by crawling landing pages to also collect and crawl URLs of subpages. Finally, we identify differences by comparing commonly used settings and technologies among these pages, shown in Fig. 2. The following sections elaborate on each process phase.

Before describing our measurement process phases, we define several terms we use throughout this work. A *website* consists of multiple web pages, and a *web page* indicates information displayed in a browser when accessing a URL. Furthermore, we separate web pages into *landing pages* and *subpages*. The former are web pages corresponding to URLs, for example, of a root directory and of "official sites". The latter are web pages with the same domain name (more precisely, FQDN excluding its subdomains) of the landing page but different URL paths.

4.1 URL Collection and Crawling

To measure the web security governance of companies, we must exclude URLs of personal websites, such as blog websites, from measurement targets. Therefore, we collected domain names listed on Alexa Top Sites [25], which provides a website ranking based on page views. The URLs from the list are used as landing pages.

Subpages are chosen based on the landing pages. In practice, we collected URLs of subpages through search results of the landing pages' domain names. Note that we excluded non-html pages, e.g., PDF file pages, and used internal links from landing pages to bolster our subpages if the search results were insufficient.

To collect web data for the next phase, i.e., governance analysis, we access the URLs of landing and subpages using a browser. The web data includes URLs, HTTP headers, HTML content, JavaScript, and cookies that can be collected by accessing the input URL.

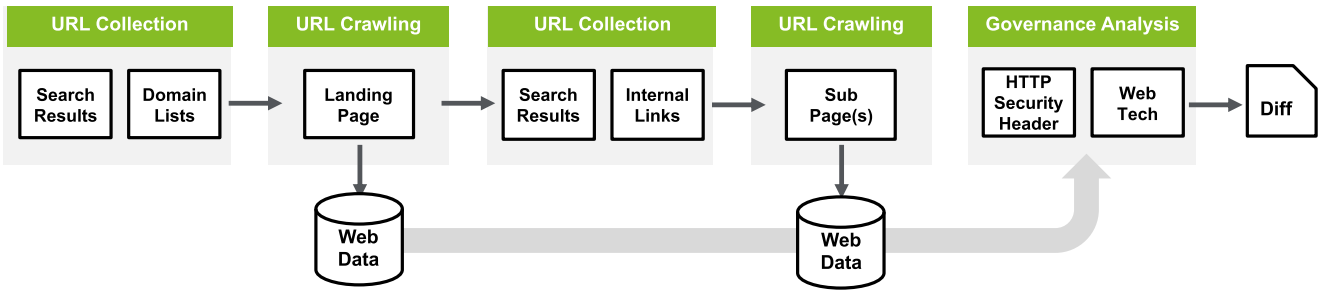

Fig. 1 Measurement Process of Web Security Governance

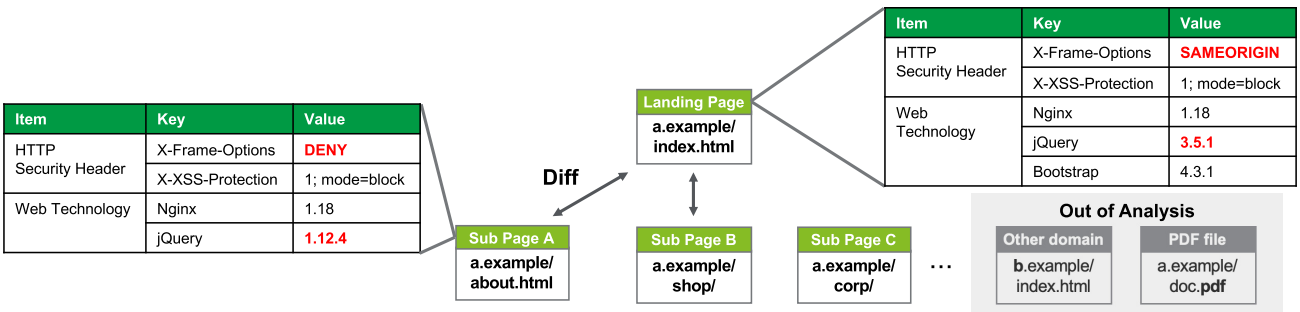

Fig. 2 Analysis of Web Security Governance

4.2 Governance Analysis

We identify differences in web security measures among landing and subpages based on collected data. As web security indicators, we use the settings of HTTP security headers and versions of web technologies as described in Sect. 2.1. In Fig. 2, our analysis identifies the `X-Frame-Options` header value and the jQuery version as differences between the landing page and subpage A.

4.2.1 Analysis of HTTP Security Headers

HTTP security headers are also used for security indicators because of their characteristics. In particular, five response headers, namely `Strict-Transport-Security`, `Content-Security-Policy`, `X-Frame-Options`, `X-XSS-Protection`, and `X-Content-Type-Options`, were studied for their popularity and statistics by many researchers [8], [13], [14], [16], [18]. Our analysis also detects these headers as security indicators and compares the values used on landing and subpages.

4.2.2 Analysis of Web Technologies

We can detect web technologies, such as servers and applications, and their versions with two methods: Static analysis and dynamic analysis [9].

(1) Static analysis.

This method detects technologies used in a website by matching keywords and regular expressions to URLs, HTTP headers, and HTML bodies. The versions can be extracted from pattern matching results.

(2) Dynamic analysis.

This method detects technologies used in a website by analyzing data in the global scope of JavaScript. The versions can be extracted from specific variables or objects defined in the scope.

Our analysis detects web technologies using the above two methods and compares these version values of technologies used on landing and subpages. Note that these technologies are detected on the web pages of not only first-parties but also third-parties.

4.3 Measurement Setup

To build a URL list of landing pages, we used root-level URLs of domain names that were listed in the top 10,000 Alexa rankings on November 11st, 2020. We collected 1 URL of landing pages and up to 10 URLs of subpages using internal links of the landing page. Note that we removed domain aliases[†] (a total of 694 domain names were removed) using Mozilla Public Suffix [26] and `tldextract` [27].

To collect web data of each page, we used a Chromium browser [28] on Ubuntu. Since JavaScript executions and asynchronous communications will occur after loading web content, we forced the browser to wait for the network to idle while crawling. Note that if content loading did not complete within three minutes, we timed out the access. HTTP

[†]Domain aliases are domains with the same second-level domain, but different top-level domains, e.g., *google.com* and *google.co.jp*.

response headers of input or redirect URLs were used to detect HTTP security headers. To detect web technologies, we used Wappalyzer [29], which can detect 1,800 technologies across 69 categories, such JavaScript libraries, web servers, and CMS [30], by the methods described in Sect. 4.2.2.

5. Measurement Results

5.1 Results of URL Collection and Crawling

We show results of crawling URLs of landing and subpages in December 2020 in Table 2. 8,190 landing pages responded with content successfully. The others were network errors (e.g., DNS errors and timeout errors) or server-side errors (e.g., HTTP400s and HTTP 500s). In addition to the landing pages, 74,608 URLs of subpages were collected based on the domain names of landing pages and among these, 69,409 responded with content successfully. We analyze landing pages and subpages without network and server-side errors in the following sections.

5.2 Results of Governance Analysis

5.2.1 Differences Among Multiple Pages

We analyzed differences in setting values of HTTP security headers and version values of web technologies among multiple pages. As a result, 4,680 (57.1%) websites had differences. Table 3 shows that almost all of these differences were versions of web technologies (more details in Sect. 5.2.3), and that settings of security headers only accounted for 447 differences (more details in Sect. 5.2.2). In addition, there were 2,812 (34.3%) websites with "no difference" and 698 (8.5%) websites with only landing pages and "no subpages". Websites with only links to different subdomains or PDF contents as well as websites without search results and internal links were classified as "no subpage", since the target of our analysis were subpages with the same domain name as the landing pages and HTML contents. In the following sections, we elaborate on differences in HTTP security headers and web technologies of both lists together.

Table 2 Results of URL Crawling

Result	Landing Page	subpage
HTTP 200 OK	8,190	69,409
Network error	818	3,731
Server-side error	257	1,468
Total	9,306	74,608

Table 3 Differences among landing/subpages

Difference	# of Websites	Rate (N=8,190)
Web technology	4,233	51.7%
No difference	2,812	34.3%
No subpage	698	8.5%
Both header and technology	295	3.6%
Security header	152	1.9%

5.2.2 Differences in HTTP Security Headers

Table 4 shows HTTP security headers detected on landing pages. The top detected header was the XFO header, which was detected on 1,838 (22.4%) pages. This XFO header is used for mitigating click-jacking attacks by validating content injected from other websites via frames. However, Calzavara et al. recommended using it in combination with a CSP header because it is not effective against click-jacking attacks using double framing [18]. A CSP header protects browsers from XSS and content injection attacks through resource policy as described in Table 1. Although a CSP header can mitigate a wide range of data injection attacks, it is known for its complex settings, many misconfigurations, and lack of adoption [16], [17]. Our results also show a lack of CSP adoption, with only 784 (9.6%) websites using the header.

While comparing them with the above security headers of landing pages, we show security headers with different settings detected on subpage settings in Table 5. The breakdown of these differences is as follows: "Policy Change" means that different settings were used, "Multiple Values" means multiple settings, and "Misconfiguration" means settings with extra characteristics. For example, Policy Changes were setting changes from `SAMEORIGIN` to `DENY` in `X-Frame-Options` headers and set/unset of `includeSubdomains` in `Strict-Transport-Security` headers. The headers with "Multiple Values" were, for example, "`X-XSS-Protection: 1; mode=block, 1; mode=block`", "`X-Frame-Options: SAMEORIGIN, DENY`" (this example used different values at the same time), and "`X-Frame-Options: SAMEORIGIN, SAMEORIGIN, SAMEORIGIN, SAMEORIGIN, SAMEORIGIN, SAMEORIGIN, SAMEORIGIN, SAMEORIGIN`" (this example used the same value repeated **eight times**). Calzavara et al. reported that

Table 4 HTTP security headers detected on landing pages

Header Name	# of Pages	Rate (N=8,190)
XFO	1,838	22.4%
HSTS	1,804	22.0%
XCTO	1,661	20.3%
XSSP	1,413	17.3%
CSP	784	9.6%

Table 5 HTTP security headers with different settings detected on subpages

Header Name	# of Diffs	Breakdown of Differences		
		Policy Change	Multiple Values	Misconfiguration
HSTS	181	90	63	28
XFO	155	63	84	8
CSP	145	128	6	11
XSSP	67	15	45	7
XCTO	63	1	55	7

Table 6 Top 10 software with versions detected on landing pages

Software Name	# of Pages	Rate (N=8,190)
jQuery	6,011	73.4%
Nginx	3,627	44.3%
PHP	1,949	23.8%
Microsoft ASP.NET	1,877	22.9%
Apache	1,684	20.6%
Bootstrap	1,649	20.1%
IIS	1,501	18.3%
jQuery UI	1,286	15.7%
Jetty	1,044	13.3%
OpenSSL	1,067	13.0%

Table 7 Top 10 software with different versions detected on subpages

Software Name	Category	# of Websites
Nginx	Web Server	3,077
jQuery	JavaScript Library	1,838
Apache	Web Server	1,468
Jetty	Web Server	744
Apache Traffic Server	Web Server	587
IIS	Web Server	567
PHP	Programming Language	537
Microsoft ASP.NET	Web Framework	434
Bootstrap	UI Framework	325
Varnish	Caching	242

these headers with multiple values were interpreted differently depending on the browser and lead to inconsistencies [18]. The last "Misconfiguration" were tiny mistakes such as inclusions of commas and semicolons.

We assume that the causes of these differences in HTTP security headers are manual settings of servers/applications and header manipulations by middleboxes such as network appliances [31].

5.2.3 Differences in Web Technologies

We enumerate the top 10 software with versions detected on landing pages in Table 6. The top detected software is jQuery, which is a popular JavaScript library, and has 75.9% market share in global websites as of June 2020 [32]. Our results also show that 73.4% of websites used jQuery, suggesting that jQuery is the most influential web technology.

Next, we enumerate the top 10 software with different versions detected on landing and subpages and its category† in Table 7. The top 5 software with different versions among landing/subpages were web servers and jQuery. In the following sections, we investigate the details of differences in these web servers and JavaScript libraries, i.e., Nginx and jQuery.

5.3 Differences in Web Servers

A first-party web server can be identified through the corresponding domain name. Hence, we can distinguish whether a web server belongs to a first-party or third-party domain.

†We used the categorization result from Wappalyzer.

Table 8 Breakdown of differences in Nginx versions

Breakdown of Differences	# of Diffs	Rate (N=3,077)
Use of different versions among third-party servers	2,288	74.4%
Version hiding	529	17.2%
Use of different versions among first-party and third-party servers	186	6.0%
Use of different versions among first-party servers	74	2.4%

In this section, we analyze differences in Nginx versions by focusing on their hosts.

5.3.1 Differences in Nginx Versions

We analyzed the use of Nginx in first-party and third-party domains for each web page and identified differences in Nginx versions. Table 8 shows the breakdown of the differences. The use of different versions among third-party servers accounted for over 70% of the differences. Because third-party content, such as ads and analytics, used on each page changed, the web servers that hosted the content also changed. The second difference was "version hiding" which is a popular hardening technique [14]. In addition to the differences caused by third-party dynamics, only 74 websites used different versions of Nginx in **first-party** domains. We elaborate on these first-parties in the next section.

5.3.2 Different Nginx Versions Behind Same Domain Name

We analyzed Nginx versions used for each domain name. As a result, we found Nginx versions were often changed because the domain name was the same, but IP addresses were different. The first-party domain names with such server behaviours were 159 domains. We assume that it is due to load balancing that distributes website access to multiple IP addresses. However, version changes depending on server loads might have a negative impact on the reproducibility of web security measures. Therefore, we suggest that webmasters should use the same version by applying governance on at least web servers under their control.

5.4 Differences in JavaScript Libraries

Unlike web servers (i.e., server-side technologies), we can use JavaScript libraries (i.e., client-side technologies) hosted on both first-party and third-party domains. Hence, we analyze version statistics, differences in versions, detection and inclusion methods of jQuery to measure the governance from various perspectives.

5.4.1 jQuery Versions

We visualize the results of counting and sorting jQuery versions detected on landing and subpages in Fig. 3. Figure 3 shows that the major version 1.x was detected more than

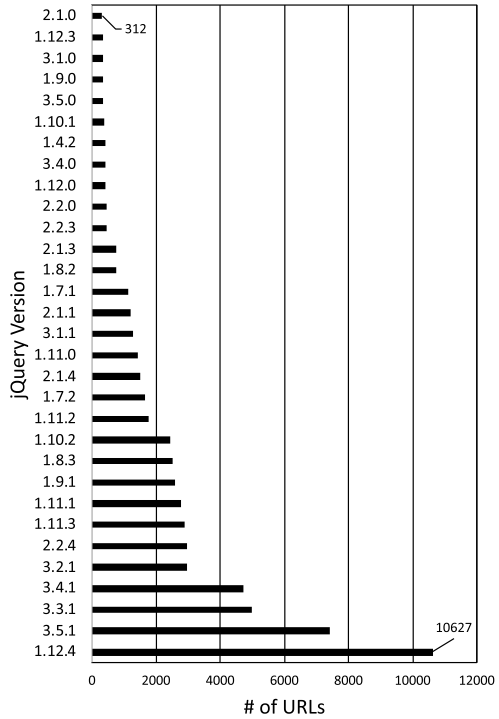

Fig. 3 Distribution of detected jQuery versions ($N > 300$)

Table 9 Pattern of differences in jQuery versions

# of Patterns	# of Websites	Rate (N=1,838)
2 patterns	1,301	70.8%
3 patterns	365	19.9%
4 patterns	106	5.8%
5 patterns	43	2.3%
6 patterns	20	1.1%
7 patterns	3	0.2%

the newer 3.x version. The most detected version, jQuery 1.12.4, is the latest release of major version 1.x as of August 2020, and it is known to be used for backward compatibility by WordPress [33]. In CMSs like WordPress, many plugins created by third-party developers use jQuery. If no backward compatibility was provided, many websites would be affected by errors caused by these plugins. We can therefore assume that these old versions continue to be used due to compatibility issues.

We found that there are 1,437 websites using multiple different versions of jQuery on the same page. This means that, for example, a landing page used jQuery with two versions, 1.6.2 and 1.8.3, at the same time. Although these duplicate library inclusions may be related to server-side templating, third-party content, or the combination of independently developed components, webmasters do not notice them unless any critical errors occur. However, duplicate library inclusions should be fixed because they lead to nondeterministic behaviour with respect to vulnerabilities and can be beneficial for long-lasting attack campaigns such as fake jQuery injections [9], [12].

Table 10 Website example with 7 patterns of jQuery versions

Page	Detected jQuery Versions
Landing page	1.11.1
subpage 1	1.10.2
subpage 2	1.11.2
subpage 3	1.2.6, 1.4.2
subpage 4	1.2.6, 1.11.1
subpage 5	1.2.6, 1.11.2
subpage 6	1.10.2, 1.12.2

Table 11 Use of jQuery with versions detected by only dynamic analysis

# of Pages	# of Websites	Rate (N = 1,838)
0 page (Not used)	521	28.3%
1 page	825	44.9%
2 pages	419	22.8%
3 pages	54	2.9%
4 pages	16	0.9%
5 pages	3	0.2%

5.4.2 Differences in jQuery Versions

To reveal whether the 1,838 differences in jQuery versions identified in Sect. 5.2.3 occurred on only one page or multiple pages, we analyzed jQuery version patterns. We count the number of patterns of jQuery versions used in 1 landing page and up to 10 subpages, i.e., up to 11 patterns, and show the results in Table 9. Surprisingly, there were 3 websites using different jQuery versions on **7 web pages**. We show the jQuery patterns in one of the three websites in Table 10. As you can see, this website used various jQuery versions and also used duplicate inclusions on subpages from 3 to 6. These websites had in common that the web contents and design of each page were totally different and that the first directory of each URL is also different. For example, these pages were a recruitment page, a teaser site, pages for branches/offices/shops, a product page, and a news page. We assume that these differences in client-side technologies occurred, for example, when switching website contexts and when webmasters are different for each page.

5.4.3 Detection Method of jQuery

Wappalyzer can detect jQuery and its version by static analysis of URLs and dynamic analysis of JavaScript. The static analysis can detect jQuery versions if URLs include version values explicitly, but cannot detect version values from URLs such as `https://a.example/lib/jquery.js`. On the other hand, dynamic analysis can detect the last loaded version of jQuery even if URLs do not contain any explicit information. Such jQuery detected by only dynamic analysis was used in 1,317 (71.7%, N=1,838) websites, and used in multiple pages as shown in Table 11. To identify whether the implicit inclusion affects jQuery version governance, we manually investigated the 73 websites that used different jQuery versions on 3 or more pages shown in Table 11. We found that some of these websites included jQuery using

Table 12 jQuery inclusion methods in websites with differences

Inclusion Method	# of Websites
First-party only (internal links)	426
Third-party only (external links)	766
Mixed first-party and third-party	400

a bundle file of multiple JavaScript libraries in addition to URLs without version information. A bundle file of libraries can be managed and generated by a module bundler, e.g., webpack [34]. However, in addition to these bundle files, some websites used other versions of jQuery, i.e., duplicate jQuery inclusions in Sect. 5.4.1. Therefore, we can infer that webmasters used multiple jQuery versions due to forgetting about a bundled jQuery or a manual jQuery inclusion.

5.4.4 Inclusion Methods of jQuery

There are two inclusion methods of jQuery. One is to use jQuery that was downloaded from official sites and package managers (e.g., npm [35], [36]) and was hosted on own (first-party) servers, and the other is to use jQuery hosted on external (third-party) servers such as those provided by Google and Microsoft [37], [38]. To identify whether differences in these inclusion methods lead to version differences, we investigated the hosts of loaded jQuery libraries.

We show the results of analyzing inclusion methods based on the URLs in Table 12. The number of jQuery inclusions hosted on third-party only was higher than first-party only. In addition, there were 400 (21.8%, N=1,838) websites that did not align with the first-party or third-party hosts of jQuery. Because the mixed inclusions were few on websites without differences of jQuery versions, only 121 (2.9%, N=4,173), it suggests that differences in inclusions methods of jQuery contribute to the differences in jQuery versions.

6. Discussion

6.1 Improvement Plans for Governance Failures

We suggest four improvement plans shown in Table 13 against governance failures identified in the previous section.

(1) Review of Guidelines.

Establishing guidelines in an organization is an effective way to govern website development and operations. We can apply web security based on the current requirements by preparing and deploying guidelines based on risk assessment results of the business, information, assets, and systems. In particular, we believe that this plan can improve governance failures by being applied to websites with different contexts and webmasters. The standardized guidelines will govern webmasters and improve the consistency of website maintenance. Since the evolution of web technology and the rapidly changing role of websites, it is also important to regularly review the guidelines to prevent them from becoming obsolete or stale [2].

Table 13 Improvement plans to governance failures identified by our measurements

Improvement Plan	Governance Failure
Review of guidelines	Difference in website's context and webmasters (Sect. 5.4.2)
Automation of web security	Multiple HTTP headers (Sect. 5.2.2) Different server versions (Sect. 5.3.2) Use of bundle library (Sect. 5.4.3) Mixed library inclusions (Sect. 5.4.4)
Supply chain management	Difficulty of managing third-party content (Sects. 5.2.3, 5.3.1, and 5.4.4) Technical limitation caused by third-party content (Sect. 5.4.1)
Regular assessment by third-parties	All of the above

(2) Automation of Web Security.

A promising improvement plan against the governance failures of manual operations is automation. This is called SecDevOps (or DevSecOps), and can automatically perform certain web security checks when updating functions or releasing systems by integrating security factors into DevOps. DevOps is a culture and practice that orchestrates development and operations for maintaining system quality and providing value to users promptly. We can automate processes such as building, testing, releasing and deploying by using tools created for DevOps [39]. SecDevOps can automate security checks by integrating logging, monitoring, secure coding and vulnerability checks into DevOps [40]. Therefore, webmasters can be aware of governance failures via automated checks of SecDevOps. Among governance failures identified by our measurements, we can expect to improve the prevention of multiple HTTP headers, different server versions, duplicate library bundles, and mixed library inclusions.

(3) Supply Chain Management.

It is important to check the use of third-party content to decrease attack surfaces stemming from unintended defects and vulnerabilities [5], [7]. Webmasters must pay attention to select third-party vendors to decrease risks caused by server-side technologies because these technologies cannot be controlled by webmasters. On the other hand, client-side technologies, such as jQuery and CMSs, can be controlled and updated by webmasters. In Sect. 5.4.1, we found compatibility issues caused by CMS plugins by detecting the use of jQuery 1.x. Webmasters must also keep plugins up to date. Since outdated CMS plugins tend to be targeted by attackers during website compromises, it is necessary to manage and control these third-party dependencies [41], [42]. Another approach to verify third-party supply chains is the use of Subresource Integrity (SRI) [43]. The SRI is a security feature that enables browsers to verify that third-party content has no unexpected manipulation by checking a

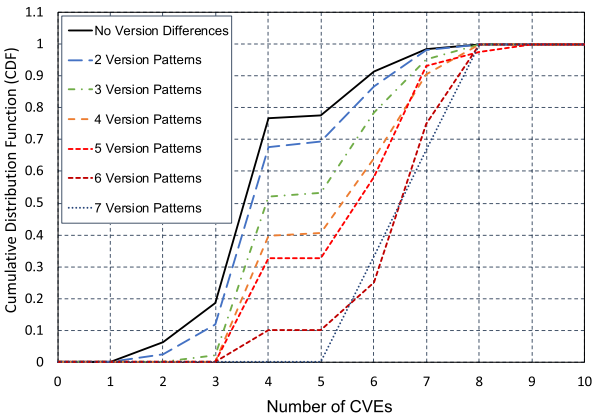

Fig. 4 Number of jQuery-related CVEs on websites without jQuery differences and websites with each version pattern

cryptographic hash of the resource. In addition to on-demand third-party content, it is also important to manage and control third-party content, such as JavaScript libraries, that is downloaded in advance. The automation tools mentioned above are suitable for management purposes [9]. For example, the JavaScript package manager npm can prevent duplicate library inclusions and manage which versions of which libraries are downloaded and used on a website [36].

In the case of entrusting website development and operations to a third-party, we should define what guidelines, tools, settings and versions to use in requirements or specifications. It is also crucial to govern third-party operators so that the same level of web security applies to systems developed by them [44], [45].

(4) Regular Assessment by Third-parties.

To prevent governance failures, security assessments by independent third-parties play a key role in addition to security checks by SecDevOps. We can evaluate the state of web security measures, meaning whether websites comply with guidelines and web security measures are sufficient, by performing our governance analysis and similar assessments regularly [44], [45].

6.2 Web Security Governance for Cyber Resilience

We can suspect that websites with governance failures have a higher risk of containing defects and vulnerabilities. We calculated the CDF (Cumulative Distribution Function) of the number of jQuery-related CVEs on websites without differences in jQuery and with each other pattern. Note that we counted the number of CVEs using public vulnerability databases [46]. Figure 4 shows that websites with differences have more CVEs than those without. Moreover, we can say that more patterns usually translate into more CVEs although results were partially reversed on websites with 7 version patterns due to the small amount of data used for analysis. In other words, it means that governing software versions leads to a decrease in the number of known vulnerabilities on websites.

Many known vulnerabilities and security indicators should not be interpreted to mean that a website is exploitable or defensible [14]. However, websites with controlled software settings and versions can more promptly respond to threats through security measures, such as updating and patching, in order to respond to zero-day vulnerabilities than websites without. Therefore, we can expect that the web security governance measured in our work can be utilized as a security indicator that measures a facet of cyber resilience.

6.3 Compatibility Issue

In our measurements, we analyzed the consistency of header settings and software versions. However, some websites have difficulty governing these settings and versions due to compatibility issues [11], [47]. Especially jQuery is often considered a problem that prevents version updates because it may lack backward compatibility even between minor versions [9]. One solution to this compatibility issue is to use the jQuery Migrate plugin [48]. This plugin provides functions that, for example, restore removed APIs and additionally show warnings when removed and/or deprecated APIs are used. The official website also provides upgrading guides for help when upgrading jQuery [49]. However, the plugin is not pervasive because it was only used in 611 (7.5%) websites and was not included in top 10 ranking in Table 6.

Although it might take a long time to fix governance failures on websites due to these compatibility issues, we suggest to make future decisions based on their role and importance. For example, it is also an option to abolish or merge these websites altogether.

6.4 Governance Including Subdomains

In our measurements, we assumed that web pages with the same domain name are managed by the same organization, and our governance analysis also identified differences among these web pages. However, there are websites using many subdomains described in Sect. 5.2.1. We can suspect that these websites including subdomains are managed by the same organization. Hence, we analyze the web security governance of subdomains to examine whether they inherited the rules of their base domains.

First, we collected one subdomain for each base domain using a search engine. We randomly extracted 500 base domains each with and without differences listed in Table 3. Note that we chose this method for focusing on subdomains used for websites although there are various other methods to collect subdomains by using DNS, certificates and WHOIS records. As a result, 424 and 410 subdomains were collected from the base domains with and without differences, respectively. Next, we analyzed the web security governance of these subdomains using the methods shown in Sect. 4. Subdomains of base domains with existing differences had further differences in 287 (72.2%) websites, and

those without existing differences had no further differences in 284 (71.7%) websites. Subdomains are generally used for creating new websites or switching design themes and contexts from base domains. Although the level that web security governance is applied at varies depending on the organization, we can say that many websites tend to inherit their adherence/non-adherence among subdomains.

6.5 Limitation

Our two approaches of web technology detection, i.e., static analysis and dynamic analysis, have cases where both approaches can fail, described in Sect. 5.4.3. For example, web technologies outside of Wappalyzer [29] detection capability and heavily modified servers and libraries cannot be detected. However, we consider that to have negligible impact on our measurement results since web servers and JavaScript libraries with high market share can be detected.

Our governance analysis compared only a landing page with the subpages because we thought that a landing page is the most maintained and should be the basis of a website. However, comparing among subpages each other is also beneficial to identify differences in settings and versions of new software that are not used in the landing page. As future work, we plan to extend our governance analysis to identify differences among subpages.

Our governance analysis did not compare the presence of HTTP security headers and web technologies, but only their set values and version values. Since the variability of these values varies depending on the headers and technologies themselves, the likelihood of these differences also changes. For instance, almost all jQuery versions are referenced in `Major.Minor.Patch` format while almost all Drupal versions are referenced in `Major` only format [50]. In addition, we cannot detect differences in web technologies without any version releases at all. Therefore, it is possible that the number of websites with governance failures is potentially higher than our measurement results indicate.

6.6 Ethical Consideration

In our measurements, we collected data by crawling multiple pages of websites. During the crawling, we distributed access and time slots to avoid overloading specific websites. Moreover, in order not to have a negative impact on specific companies and organizations, we analyzed websites without mentioning specific names.

7. Conclusion

The increasing complexity of websites makes them almost uncontrollable. To better control websites, it is important to manage and use common software settings and versions across all web pages of the website by applying security governance. In addition, applying security governance to websites can help address new vulnerabilities and threats and, as a result, improve their cyber resilience. We hope that our measurement results will trigger regular web security checks and security governance improvements to harden websites.

References

[1] J. Vijayan, "Attackers Compromised Dozens of News Websites as Part of Ransomware Campaign." https://www.darkreading.com/attacks-breaches/attackers-compromised-dozens-of-news-websites-as-part-of-ransomware-campaign/d/d-id/1338265, July 2020.
[2] NIST, "SP 800-35: Guide to Information Technology Security Services." https://doi.org/10.6028/NIST.SP.800-35, Oct. 2003.
[3] NIST, "SP 800-44: Guidelines on Securing Public Web Servers." https://doi.org/10.6028/NIST.SP.800-44ver2, Oct. 2007.
[4] ISEC IPA, "10 Major Security Threats 2019." https://www.ipa.go.jp/files/000076989.pdf, Aug. 2019.
[5] N. Nikiforakis, L. Invernizzi, A. Kapravelos, S. Van Acker, W. Joosen, C. Kruegel, F. Piessens, and G. Vigna, "You Are What You Include: Large-scale Evaluation of Remote JavaScript Inclusions," ACM SIGSAC Conference on Computer and Communications Security (CCS), 2012.
[6] M. Ikram, R. Masood, G. Tyson, M.A. Kaafar, N. Loizon, and R. Ensafi, "The chain of implicit trust: An analysis of the web third-party resources loading," The Web Conference (WWW), pp.2851–2857, 2019.
[7] D. Kumar, Z. Ma, A. Mirian, J. Mason, J.A. Halderman, and M. Bailey, "Security Challenges in an Increasingly Tangled Web," World Wide Web Conference (WWW), pp.1–8, 2017.
[8] B. Stock, M. Johns, M. Steffens, and M. Backes, "How the Web Tangled Itself: Uncovering the History of Client-Side Web (In) Security," USENIX Security Symposium, 2017.
[9] T. Lauinger, A. Chaabane, S. Arshad, W. Robertson, C. Wilson, and E. Kirda, "Thou Shalt Not Depend on Me: Analysing the Use of Outdated JavaScript Libraries on the Web," Network and Distributed System Security Symposium (NDSS), 2017.
[10] "ISO/IEC 27014:2020 Information security, cybersecurity and privacy protection – Governance of information security." https://www.iso.org/standard/74046.html, Dec. 2020.
[11] "OWASP Secure Headers Project." https://owasp.org/www-project-secure-headers/, April 2020.
[12] Jérôme Segura, "Fake jquery campaign leads to malvertising and ad fraud schemes." https://blog.malwarebytes.com/threat-analysis/2019/06/fake-jquery-campaign-leads-to-malvertising-and-ad-fraud-schemes/, June 2019.
[13] T. Van Goethem, P. Chen, N. Nikiforakis, L. Desmet, and W. Joosen, "Large-scale security analysis of the web: Challenges and findings," International Conference on Trust and Trustworthy Computing (TRUST), pp.110–126, 2014.
[14] S. Tajalizadehkhoob, T. Van Goethem, M. Korczyński, A. Noroozian, R. Böhme, T. Moore, W. Joosen, and M. van Eeten, "Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting," ACM SIGSAC Conference on Computer and Communications Security (CCS), 2017.
[15] A. King, "Analysis of the Alexa Top 1M Sites." https://blog.mozilla.org/security/2018/02/28/analysis-alexa-top-1m-sites-2/, Feb. 2018.
[16] S. Calzavara and M. Bugliesi, "Content Security Problems? Evaluating the Effectiveness of Content Security Policy in the Wild," ACM SIGSAC Conference on Computer and Communications Security (CCS), pp.1365–1375, 2016.
[17] S. Roth, T. Barron, S. Calzavara, N. Nikiforakis, and B. Stock, "Complex Security Policy? A Longitudinal Analysis of Deployed Content Security Policies," Network and Distributed System Security Symposium (NDSS), 2020.
[18] S. Calzavara, S. Roth, M. Backes, and B. Stock, "A Tale of Two Headers: A Formal Analysis of Inconsistent Click-Jacking

Protection on the Web," USENIX Security Symposium, 2020.
[19] B. Eriksson and A. Sabelfeld, "AutoNav: Evaluation and Automatization of Navigation Policies," The Web Conference (WWW), pp.1320–1331, 2020.
[20] S. Roth, M. Backes, and B. Stock, "Assessing the Impact of Script Gadgets on CSP at Scale," ACM Symposium on Information, Computer and Communications Security (AsiaCCS), 2020.
[21] M. Vasek, J. Wadleigh, and T. Moore, "Hacking is not random: a case-control study of webserver-compromise risk," IEEE Transactions on Dependable and Secure Computing, vol.13, no.2, pp.206–219, 2015.
[22] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," Computer Networks, vol.33, no.1, pp.309–320, 2000.
[23] K. Soska and N. Christin, "Automatically Detecting Vulnerable Websites Before They Turn Malicious," USENIX Security Symposium, 2014.
[24] T. Urban, M. Degeling, and T. Holz, "Beyond the Front Page: Measuring Third Party Dynamics in the Field," The Web Conference (WWW), 2020.
[25] "Alexa Top Sites." https://aws.amazon.com/jp/alexa-top-sites/, June 2020.
[26] Mozilla, "Public Suffix List." https://publicsuffix.org/, 2021.
[27] J. Kurkowski, "tldextract." https://pypi.org/project/tldextract/, 2021.
[28] "The Chromium Projects." https://www.chromium.org/Home, Feb. 2020.
[29] E. Alias, "Wappalyzer." https://www.wappalyzer.com/, 2021.
[30] E. Alias, "Technologies - Wappalyzer." https://www.wappalyzer.com/technologies/, 2021.
[31] G. Tyson, I. Castro, F. Cuadrado, and S. Uhlig, "Exploring HTTP Header Manipulation In-The-Wild," World Wide Web Conference (WWW), 2017.
[32] W3Techs, "Usage statistics of JavaScript libraries for websites." https://w3techs.com/technologies/overview/javascript_library, May 2020.
[33] "Why wordpress only use old jQuery version is 1.12.4?." https://wordpress.org/support/topic/why-wordpress-only-use-old-jquery-version-is-1-12-4/, 2021.
[34] "webpack." https://webpack.js.org/, 2021.
[35] "jQuery CDN." https://code.jquery.com/, 2021.
[36] "npm." https://www.npmjs.com/, 2021.
[37] "Hosted Libraries | Google Developers." https://developers.google.com/speed/libraries, 2021.
[38] "Microsoft Ajax Content Delivery Network." https://docs.microsoft.com/en-us/aspnet/ajax/cdn/overview, 2021.
[39] C. Ebert, G. Gallardo, J. Hernantes, and N. Serrano, "DevOps," IEEE Softw., vol.33, no.3, pp.94–100, 2016.
[40] V. Mohan and L. Ben Othmane, "SecDevOps: Is it a marketing buzzword? Mapping research on security in DevOps," International Conference on Availability, Reliability and Security (ARES), pp.542–547, 2016.
[41] D. Fiser, "Looking into Attacks and Techniques Used Against WordPress Sites." https://blog.trendmicro.com/trendlabs-security-intelligence/looking-into-attacks-and-techniques-used-against-wordpress-sites/, Dec. 2019.
[42] D. Saunders, "CMS attacks on the rise." https://hello.global.ntt/en-us/insights/blog/cms-attacks-on-the-rise, June 2020.
[43] Mozilla, "Subresource Integrity." https://developer.mozilla.org/en-US/docs/Web/Security/Subresource_Integrity, Dec. 2020.
[44] NIST, "SP 800-30: Guide for Conducting Risk Assessments." https://doi.org/10.6028/NIST.SP.800-30r1, Sept. 2012.
[45] NIST, "SP 800-37: Risk Management Framework for Information Systems and Organizations: A System Life Cycle Approach for Security and Privacy." https://doi.org/10.6028/NIST.SP.800-37r2, Nov. 2018.
[46] NIST, "NVD Data Feeds." https://nvd.nist.gov/vuln/data-feeds, 2021.
[47] T. Dumitraş and P. Narasimhan, "Why do upgrades fail and what can we do about it? toward dependable, online upgrades in enterprise system," Proc. 10th ACM/IFIP/USENIX International Conference on Middleware, 2009.
[48] "jQuery Core Upgrade Guides." https://jquery.com/download/\#jquery-migrate-plugin, 2021.
[49] "jQuery Migrate Plugin." https://jquery.com/upgrade-guide/, 2021.
[50] C. Dresen, F. Ising, D. Poddebniak, T. Kappert, T. Holz, and S. Schinzel, "CORSICA: Cross-Origin Web Service Identification," ACM Symposium on Information, Computer and Communications Security (AsiaCCS), 2020.

Yuta Takata received his B.E., M.E., and Ph.D. degrees in computer science and engineering from Waseda University, Japan in 2011, 2013, and 2018. He was a researcher at NTT from 2013 to 2018. He is currently a senior researcher and a manager at Deloitte Tohmatsu Cyber LLC, Tokyo, Japan. Since joining Deloitte in 2019, he has been engaged in R&D of technologies and solutions related to cyber security, web security and privacy while working on utilizing the research results in business.

Hiroshi Kumagai worked as a lead analyst in JPCERT/CC from 2011 to 2015. He was a researcher at PwC from 2015 to 2019. In 2019, he joined Deloitte Tohmatsu Cyber LCC, Tokyo, Japan where he is currently a principal researcher. His research interests include threat intelligence, dark web, cryptocurrency, fake news, and he has been engaged in R&D of technologies and solutions based on these interests.

Masaki Kamizono led the Cyber Security Laboratory at PwC and worked as a senior researcher at NICT from 2015 to 2019. In 2019, he joined Deloitte Tohmatsu Cyber LCC, Tokyo, Japan as CTO to launch the Advanced Cyber Security Laboratory. He leads the R&D team, and consistently develops new solutions and new businesses based on R&D. He has also been engaged in human resource development.