A Case-Control Study on the Server-Side Bandages Against XSS

JUKKA RUOHONEN and VILLE LEPPÄNEN, University of Turku

This paper surveys the server-side use of security-related options for protecting websites against cross-site scripting (XSS) attacks. By using data from a bug bounty platform, the use of these header-based options is approached with a case-control study that contrasts popular Internet domains against less popular domains that have explicitly been verified to have been vulnerable to XSS. According to the results based on the analysis of nearly 800 thousand domains, (a) the header-based security options are only infrequently used. However, (b) the domains known to have been vulnerable to XSS have been much less likely to use these options compared to popular domains. Furthermore, (c) the options surveyed tend to statistically form clear latent dimensions, which can be speculated to relate to the effort required to enforce strict security policies for websites.

1. INTRODUCTION

Cross-site scripting vulnerabilities have continued to frequently appear in the top-rankings of the most common security bugs in web applications [OWASP 2018]. The essence behind these security bugs is simple: an attacker injects malicious executable code to a vulnerable website, and a client's web browser executes this code during rendering of the website. The consequences for clients may range from stealing of web cookies and browsing history to phishing and even key logging.

If time, resources, and talent are all available, the "best strategy for protecting webservers is to write secure Web applications from scratch" [Stritter et al. 2016]. Combined with the commonplace lack of time, resources, and talent, the sheer complexity and the chaotic nature of the current Web have prompted the introduction of many countermeasures against XSS. Most of these are bandages that merely patch the symptom—the only real remediation against cross-site scripting is arguably proper input validation. However, as it is deviously difficult to do this in practice [Weinberger et al. 2011], different remediation solutions have been considered practically on all sides of the current Web [Stritter et al. 2016]. On the infrastructure side, remediation is deep-rooted in the fundamental issues affecting web standards, encryption of hypertext transfer protocol (HTTP) traffic, the global public key infrastructure, the domain name system, the so-called same-origin policy, and ultimately the whole client-server paradigm. On the client-side, the mitigation solutions include sandboxes, blacklists, and different heuristics to profile the execution of JavaScript. On the server-side, the remediation solutions include web application firewalls, algorithmic solutions, and different options that can be set in HTTP header responses for instructing browsers about intended behavior. Although much of existing research has focused on the detection and prevention of XSS vulnerabilities with algorithms and heuristics [Hydara et al. 2015], the header options are noteworthy for a couple of important reasons.

First, many of these satisfy the desirable property of bandages: these are easy to apply to fix bleeding in existing implementations, and these are recognized by most browsers due to (implicit) stan-

Authors' addresses: University of Turku, FI-20014 Turun yliopisto, Finland, juanruo@utu.fi, ville.leppanen@utu.fi

In: Z. Budimac (ed.): Proceedings of the SQAMIA 2018: 7th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications, Novi Sad, Serbia, 27–30.8.2018. Also published online by CEUR Workshop Proceedings (http://ceur-ws.org, ISSN 1613-0073)

dardization [Ross et al. 2013; W3C 2016; 2017]. Second, these have enlarged the scholarly intersection between web application security research and the increasingly popular large-scale Internet measurement research [Lekies et al. 2013; Tajalizadehkhoob et al. 2017; Vasek et al. 2016; Weissbacher et al. 2014]. This intersection is also the area to which this paper contributes. The contribution stems from the data used. While numerous studies have collected data based on blacklists and related collections about domains and websites associated with spam, phishing, and malware [Ruohonen et al. 2016; Tajalizadehkhoob et al. 2017; Vasek et al. 2016], these are only implicitly related to web vulnerabilities. In contrast, this paper uses data from a so-called bug bounty specialized to the discovery and disclosure of XSS vulnerabilities in particular. In other words, a part of the data used is explicitly about websites that have been verified to have been vulnerable to security bugs the headers surveyed are meant to partially address. Equipped with this improvement to the always difficult ground truth problem, the paper sets to answer to the following three research questions (RQs):

—RQ_1: *How common is the use of HTTP headers for XSS prevention?*

—RQ_2: *Are different HTTP header fields used in conjunction with other fields designed to mitigate XSS?*

—RQ_3: *Is there a difference in the use of XSS-related HTTP header fields between popular Internet domains and less popular domains that have been vulnerable to XSS according to data from a bug bounty?*

To answer to these three questions, the paper proceeds by introducing the materials and methods in Section 2. The empirical results are presented in Section 3 and further discussed in the final Section 4.

2. MATERIALS AND METHODS

The following discussion will outline the headers surveyed, the empirical materials, and the methods used.

2.1 Header Fields

The header fields surveyed are listed in Table I. The listing is not comprehensive; only well-known fields either explicitly or at least implicitly related to XSS are surveyed. Even with this narrowing of the scope, it is not easy to demarcate between fields related to cross-site scripting and those not related to it. In other words: in many attack scenarios, XSS can be used to facilitate other attacks, or cross-site scripting can be facilitated by other attacks. The header fields surveyed can be further elaborated as follows:

—The `HttpOnly` field can be set for cookies to prevent these from being accessible by scripts executed by a client's browser [Ruohonen and Leppänen 2017b]. Setting this flag is generally recommended for session cookies in order to mitigate theft of session information via XSS.

—A value `nosniff` can be set to prevent some browsers from deducing about media types and file formats (i.e., MIME types) that do not match the declared `Content-Type`. The rationale for sniffing MIME types relates to the lack of universally accepted meta-data for web content and the common occurrence of incorrectly defined types. Unfortunately, this sniffing exposes XSS attacks [Barth et al. 2009].

—`X-Frame-Options` are not directly related to XSS *per se*, although the so-called clickjacking can be used in conjunction with XSS, cross-site request forgery (CSRF), and related attacks [Kim and Kim 2015; Takamatsu and Kono 2014]. The values specified for `X-Frame-Options` can mitigate some of these attacks by either preventing or restricting a client's browser from displaying web content through `<frame>` and `<iframe>` elements [Ross et al. 2013]. The three available values forbid framing

altogether, restrict it to the same origin than the content itself, or allow framing only from explicitly specified origins.

—*Referrer-Policy* governs the information a client sends along with the `Referer` header to the landing page. A certain class of XSS attacks are known to be possible with the `Referer` information [Gremwell BVBA 2010], although privacy and other security concerns (including CSRF) have been more pressing [Barth et al. 2008; W3C 2017]. Some of these attacks and related concerns can be remedied by forbidding referrer information to be sent with a `no-referrer` value. In addition, more fine-grained control is possible with other values [W3C 2017]. The metric labeled REFERRER-ORIGIN groups values that tighten the default client-side policy, excluding the strictest `no-referrer` policy.

—`X-XSS-Protection` is a HTTP response header field that can be set to instruct some web browsers to enable either sanitization or filtering of reflected XSS attacks [Mozilla Foundation et al. 2017]. The main targets of the field are inline resources, such as inline event handlers and inline `<script>` or `<style>` elements. In general, this field has been augmented by the more comprehensive *Content-Security-Policy*.

—*Content-Security-Policy* (CSP) is currently the most comprehensive header-based mitigation solution against cross-site scripting. The basic idea behind CSP is to use a policy document for listing those domains that are allowed to execute scripts, load web resources, or embed cross-domain functionality [W3C 2016]. The explicitly defined policy directives are grouped into the metric CSP-SRC. The metric CSP-URI, in turn, captures the presence of policy restrictions based on uniform resource identifiers. Reflecting CSP's complexity, there are numerous additional directives that can be set for further restrictions. These are grouped into CSP-MISC. Finally, it is worth remarking that the deprecated `reflected-xss` has similar functionality to `X-XSS-Protection`.

Table I. Metrics and Header Fields

Field	METRIC	Description / operationalization
Set-Cookie	HTTPONLY	⋆ A cookie with `HttpOnly`.
X-Content-Type-Options	NOSNIFF	⋆ A value `nosniff` is present.
X-Frame-Options	X-FRAME-DENY	⋆ `X-Frame-Options: deny` is set.
	X-FRAME-SAMEO	⋆ `X-Frame-Options: sameorigin` is set.
	X-FRAME-AFROM	⋆ `X-Frame-Options: allow-from` is set.
Referrer-Policy	REFERRER-NO	⋆ A value `no-referrer` is present.
	REFERRER-ORIGIN	⋆ A value `origin`, `same-origin`, `strict-origin`, `origin-when-cross-origin`, or `strict-origin-when-cross-origin` is defined.
X-XSS-Protection	X-XSS-FILTER	⋆ `X-XSS-Protection: 1` is present.
	X-XSS-BLOCK	⋆ `X-XSS-Protection: 1; mode=block` is present.
Content-Security-Policy	CSP-SRC	⋆ A value `default-src`, `script-src`, `object-src`, `style-src`, `img-src`, `media-src`, `frame-src`, `child-src`, `font-src`, `connect-src`, or `manifest-src` is defined.
	CSP-URI	⋆ A `base-uri`, `form-action`, or `frame-ancestors` is set.
	CSP-MISC	⋆ A value `sandbox`, `script-nonce`, `referrer`, `plugin-types`, `block-all-mixed-content`, or `upgrade-insecure-requests` is present.
	CSP-XSS	⋆ A CSP contains `reflected-xss`.

Much of the current XSS-related standardization work concentrates on CSP. Because the current (v.3) revision [W3C 2016] is detailed and complex, applying a strict CSP successfully requires careful fine-tuning on per-site basis. Due to the complexity, it should be also stressed that the four simple metrics in Table I only probe CSP's surface by counting the policy directives themselves; the actual

values specified for these directives differentiate a secure policy from an insecure one [Liu et al. 2016; Weissbacher et al. 2014].

2.2 Data

The dataset is based on two sources. The first source is the Alexa's conventional list of top-million most popular Internet domains [Alexa Internet, Inc. 2017]. Although generalizability remains an open question, Alexa's popularity lists have been the *de facto* benchmark sources for different Internet measurement studies [Lekies et al. 2013; Liu et al. 2016; Ruohonen and Leppänen 2017a; Tyson et al. 2017; Weissbacher et al. 2014]. The second source comes from the web vulnerabilities disseminated through the Open Bug Bounty (OBB) [2017] platform. This bug bounty is a community-based platform for dissemination of XSS and CSRF vulnerabilities in particular [Ruohonen and Allodi 2018]. Unlike some other bug bounties, OBB neither pays for vulnerabilities nor targets some particular vendors or websites.

Given the two data sources, the retrieval of the headers was implemented by following the simple scheme illustrated in Fig. 1. Before continuing further, it should be noted that a success is defined as a query that returned a status code starting either with a number 2 or with a number 3. A failure, in turn, includes all other status codes as well as queries failed for undefined or unknown reasons. Despite of these disqualification criteria, it should be emphasized that initial redirections (status code 301) were followed during the retrieval.

Each domain in both samples was queried three times in order to rule out timeouts, domain name resolution failures, and other temporary errors. Although OBB provides uniform resource locators (URLs) pointing to the vulnerable web applications, Alexa's list only contains domain names. To maintain a uniform retrieval scenario for both samples, also the entries in the OBB sample were queried based on the host field in the URLs (hence, a domain that has been exposed to multiple vulnerabilities is counted only once). The host field was further restricted to cover only fully qualified domain names. In other words, Internet protocol (IPv4) addresses were excluded during pre-processing of the OBB sample. Like in previous empirical research [Tyson et al. 2017], the actual retrieval was done with the conventional HTTP/1.1 via which the headers were retrieved based on GET requests. If an HTTP query failed, however, another GET request was attempted over the transport layer security protocol. If both requests failed for three times for a given domain during the retrieval in November 2017, the domain was finally excluded from the empirical analysis. This retrieval scenario could be further extended (for instance, by querying the domains also with `www`-prefixes; see Weissbacher14), but some limitations would still be present [Ying and Li 2016]. While acknowledging the imperfection of the retrieval routine, more than enough data was retrieved from both sources (see Fig. 2).

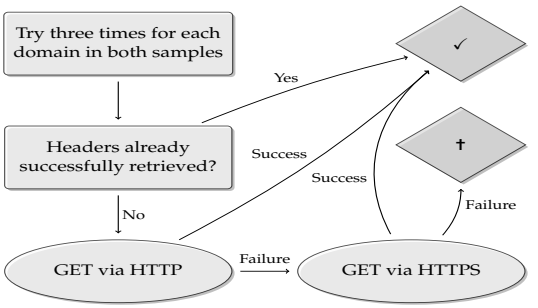

Fig. 1. A Simple Routine for Header Retrieval

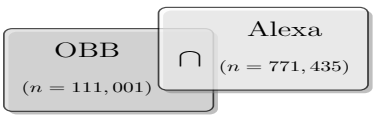

Fig. 2. Case (OBB), Control (Alexa), and Headers (n)

Finally, a few words are required about the parsing required for the metrics listed in Table I. The header fields are delivered as case-insensitive key-value pairs. As the details for standards-compliant parsing are fairly detailed, the headers were parsed line-by-line with simple string searches after both the keys and the values had been transformed to lower-case letters. As an example: to construct the X-XSS-BLOCK metric, a character `1` and a string `block` were searched from the value delivered via the transformed `x-xss-protection` key. These heuristic searches avoid some ambiguities (such as those related to semicolons and white-space). Nevertheless, it should be remarked that some outlying domains observed may be interpreted as having set an option—even though actual web browsers may interpret the option as incorrectly specified.

2.3 Methods

The following three methods are used:

(1) To answer to RQ_1, the first "method" simply plots the relative share of all metrics enumerated in Table I. For this plotting, a symmetric difference (i.e., a union without an intersection) of the two sources is used. To determine the intersection, Alexa's popularity ranks are used. A domain is classified as belonging to both sets in Fig. 2 when: (a) the domain is present in both samples, or (b) a popularity rank is provided for the domain indirectly through OBB. This twofold detection is necessary because OBB provides its own historical Alexa-ranks [Ruohonen and Allodi 2018]. In other words, some of the domains in the OBB sample may have appeared in some older ranking that is no longer publicly available from Alexa.

(2) To answer to RQ_2, the principal component analysis (PCA) is used by examining whether the metrics can be reduced into smaller dimensions. As all metrics have the same scale, a sample covariance matrix is used as the input to PCA, and the number of components is approximated with a so-called scree plot (for the details refer to Zuur et al. [2007], i.a.). The symmetric difference is again used for the two samples.

(3) To answer to RQ_3, a small case-control study is computed. Case-control studies are classical ways to conduct experimental studies in fields such as medicine and epidemiology [Rundle et al. 2012], but these have recently gained traction also in security research [Vasek et al. 2016]. By removing the intersection in Fig. 2, the case is composed of the vulnerable domains in the OBB subsample, while the domains in the Alexa subsample constitute the control. Due to the definition given for the intersection of the two samples, the case refers also to less popular domains, which are compared against the control group of popular domains. Given the $1 \leq i \leq 13$ metrics, the resulting case-control setup is illustrated in Table II.

Table II. The Case-Control Setup (frequencies n_{ij})

	OBB	Alexa	Odds ratio
$\mathrm{METRIC}_i = 1$ (present)	n_{11}	n_{12}	$\frac{n_{11} \times n_{22}}{n_{12} \times n_{21}}$
$\mathrm{METRIC}_i = 0$ (absent)	n_{21}	n_{22}	

Thus: if the nonintersecting OBB and Alexa subsamples do not differ in terms of the i:th metric, the corresponding odds ratio equals unity. A value smaller (larger) than one indicates that the vulnerable OBB domains are less (more) likely to have had a header-based XSS protection in place compared to the control domains. Given the unequal sample sizes (see Fig. 2), a simple resampling computation [Rundle et al. 2012] is used by drawing 1000 random subsets from the Alexa subsample, the size of each subset equaling $n_{11} + n_{21}$. Arithmetic mean is used to report the odds ratios from the thousand draws.

3. RESULTS

The answer to RQ_1 is clear: the HTTP header fields related to security are only infrequently used. The most common use case is to restrict the accessibility of session cookies (see Fig. 3). Roughly around one tenth of the domains observed have also restricted MIME sniffing, enforced the same-origin policy for framing, and adopted *X-XSS-Protection*. The use of CSP is strikingly rare.

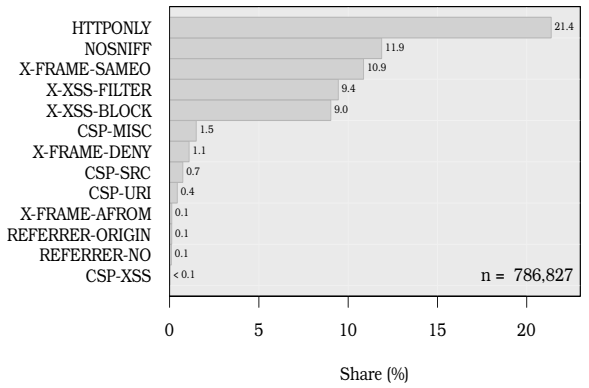

Fig. 3. Relative Share of Header Field Metrics

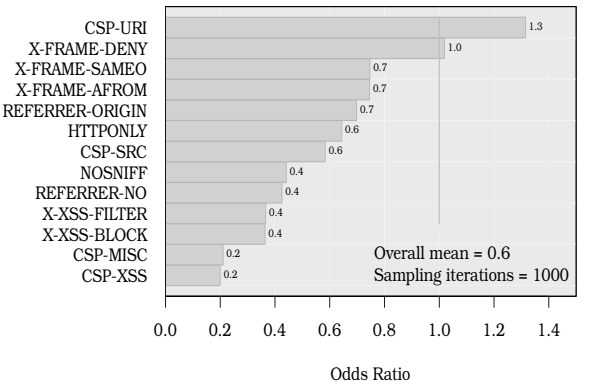

Fig. 4. Odds Ratios

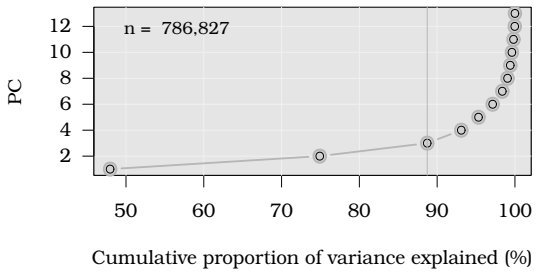

Fig. 5. A Scree Plot (PCA)

	Principal component (PC)		
	PC1	PC2	PC3
HTTPONLY		0.94	
X-XSS-FILTER	-0.51		
X-XSS-BLOCK	-0.49		
NOSNIFF	-0.54		
X-FRAME-SAMEO			-0.93

Fig. 6. Loadings (≥ 0.4 or ≤ -0.4 shown for 3 PCs)

The top-5 metrics in Fig. 3 explain as much as about 90% of the total variation in the sample (see Fig. 5). When looking at the relevant PCA loadings (see Figure 6), it seems that there are three latent dimensions: cookie restriction (PC2), XSS protection (PC1), and partial enforcement of the same-origin policy (PC3). The answer to RQ_2 is positive: the easily enforced XSS restrictions are used in conjunction with the other fields requiring little effort. The answer to RQ_3 is also clear: the less popular OBB domains (without Alexa's ranks) have been much less likely to use HTTP header fields for security improvements. The odds ratios are equal or greater than one only for two metrics (see Fig. 4), which are only infrequently used in general (see Fig. 3). The fields explicitly related to cross-site scripting—including NOSNIFF and the `X-XSS-Protection` options—attain odds ratios clearly smaller than unity. Thus, when compared to popular domains, the less popular domains that have explicitly been known to be vulnerable to XSS have failed to enforce basic protections—or, alternatively, these domains may have been vulnerable due to the lack of the protections.

4. DISCUSSION

This paper surveyed the use of HTTP header options designed to mitigate XSS and related attacks. According to the results, the use of these headers is still at a modest level even among popular Internet domains (RQ_1). This result supports recent empirical observations; even simple techniques such as JavaScript-restricted cookies and *X-XSS-Protection* are only infrequently used on the server-side of the current Web [Ruohonen and Leppänen 2017b; Tajalizadehkhoob et al. 2017]. However, not all websites are equal.

The results clearly show that less popular domains that have been verified to have been vulnerable to XSS are much less likely to use header-based protections (RQ_3). Although the statistical computations are exceptionally clear in this regard, this result should be still taken only tentatively. The reason for this caution is well-known. The most challenging part in the design of a case-control study is the selection of a control group [Rundle et al. 2012; Vasek et al. 2016]. Even with the careful subsetting of the two subsamples, the Alexa's top-million list can hardly be taken as a bulletproof control group. Furthermore, the results are only approximations because it is impossible to deduce whether the domains in the OBB subsample used header-based mitigation at the time when the vulnerabilities were disclosed through the bug bounty [Ruohonen and Allodi 2018]. A third limitation relates to the sampling strategy (see Fig. 1), which only probes a domain's main page [Ying and Li 2016]. While acknowledging these three limitations—which largely apply also to many related Internet measurement studies in general, the case-control study presented demonstrates that data from bug bounties can sharpen the assumptions about vulnerable websites.

Finally, the results further show that the XSS-related header fields tend to statistically parcel into distinct latent dimensions (RQ_2). If these dimensions are interpreted to reflect effort [Tajalizadehkhoob et al. 2017], it seems that the dimensions that require little effort are more frequently enforced. A plausible explanation relates to the ever increasing amount of dependencies to external websites [Ruohonen et al. 2018]. As the empirical results also support the existing observations about CSP's adoption problems [Weissbacher et al. 2014], it seems fair to conclude with a remark that web developers may have difficulties to implement and enforce security policies when third-party domains or URLs must be explicitly stated.

REFERENCES

Alexa Internet, Inc. 2017. The Top Million Websites. (2017). Data feed retrieved in November: http://s3.amazonaws.com/alexa-static/top-1m.csv.zip.

Adam Barth, Juan Caballero, and Dawn Song. 2009. Secure Content Sniffing for Web Browsers, or How to Stop Papers from Reviewing Themselves. In *Proceedings of the 30th IEEE Symposium on Security and Privacy (S&P 2009)*. IEEE, Berkeley, 360–371.

Adam Barth, Collin Jackson, and John C. Mitchell. 2008. Robust Defenses for Cross-Site Request Forgery. In *Proceedings of the 15th ACM conference on Computer and Communications Security (CCS 2008)*. ACM, Alexandria, 75–88.

Gremwell BVBA. 2010. Exploiting Cross-Site Scripting in Referer Header. (2010). Available online in January 2018: https://www.gremwell.com/exploiting_xss_in_referer_header.

Isatou Hydara, Abu Bakar Md. Sultan, Hazura Zulzalil, and Novia Admodisastro. 2015. Current State of Research on Cross-Site Scripting (XSS) – A Systematic Literature Review. *Information and Software Technology* 58 (2015), 170–186.

Daehyun Kim and Hyoungshick Kim. 2015. Performing Clickjacking Attacks in the Wild: 99% are Still Vulnerable!. In *Proceedings of the 1st International Conference on Software Security and Assurance (ICSSA 2015)*. IEEE, Suwon, 25–29.

Sebastian Lekies, Ben Stock, and Martin Johns. 2013. 25 Million Flows Later: Large-Scale Detection of DOM-Based XSS. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security (CCS 2013)*. ACM, Berlin, 1193–1204.

Shukai Liu, Xuexiong Yan, Qingxian Wang, and Qi Xi. 2016. A Systematic Study of Content Security Policy in Web Applications. *Security and Communication Networks* 9, 16 (2016), 3570–3584.

Mozilla Foundation and others. 2017. X-XSS-Protection. (2017). Available online in January 2018: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection.

Open Bug Bounty (OBB). 2017. https://www.openbugbounty.org/. (2017).

OWASP. 2018. Top 10-2017 Top 10. (2018). The Open Web Application Security Project (OWASP). Available online in October 2017: https://www.owasp.org/index.php/Top_10-2017_Top_10.

David Ross, Tobias Gondrom, and Thames Stanley. 2013. HTTP Header Field X-Frame-Options (RFC 7034). (2013). Internet Engineering Task Force (IETF). Available online in January 2018: https://www.ietf.org/rfc/rfc7034.txt.

Andrew Rundle, Habibul Ahsan, and Paolo Vineis. 2012. Better Cancer Biomarker Discovery Through Better Study Design. *European Journal of Clinical Investigation* 42, 12 (2012), 1350–1359.

Jukka Ruohonen and Luca Allodi. 2018. A Bug Bounty Perspective on the Disclosure of Web Vulnerabilities. In *Proceedings of the 17th Annual Workshop on the Economics of Information Security (WEIS 2018)*. Innsbruck, 1–14. URL: https://weis2018.econinfosec.org/wp-content/uploads/sites/5/2018/05/WEIS_2018_paper_33.pdf.

Jukka Ruohonen and Ville Leppänen. 2017a. How PHP Releases Are Adopted in the Wild?. In *Proceedings of the 24th Asia-Pacific Software Engineering Conference (APSEC 2017)*. IEEE, Nanjing, 71–80.

Jukka Ruohonen and Ville Leppänen. 2017b. Whose Hands Are in the Finnish Cookie Jar?. In *Proceedings of the European Intelligence and Security Informatics Conference (EISIC 2017)*. IEEE, Athens, 127–130.

Jukka Ruohonen, Joonas Salovaara, and Ville Leppänen. 2018. Crossing Cross-Domain Paths in the Current Web. In *Proceedings of the 16th Annual Conference on Privacy, Security and Trust (PST 2018)*. IEEE, Belfast.

Jukka Ruohonen, Sanja Šćepanović, Sami Hyrynsalmi, Igor Mishkovski, Tuomas Aura, and Ville Leppänen. 2016. Correlating File-Based Malware Graphs Against the Empirical Ground Truth of DNS Graphs. In *Proceedings of the 10th European Conference on Software Architecture Workshops (ECSAW 2016)*. ACM, Copenhagen, 30:1 – 30:6.

Benjamin Stritter, Felix Freiling, Hartmut König; René Rietz, Steffen Ullrich, Alexander von Gernler, Felix Erlacher, and Falko Dressler. 2016. Cleaning up Web 2.0's Security Mess—at Least Partly. *IEEE Security & Privacy* 14, 2 (2016), 48–57.

Samaneh Tajalizadehkhoob, Tom van Goethem, Maciej Korczyński, Arman Noroozian, Rainer Böhme, Tyler Moore, Wouter Joosen, and Michel van Eeten. 2017. Herding Vulnerable Cats: A Statistical Approach to Disentangle Joint Responsibility for Web Security in Shared Hosting. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS 2017)*. ACM, Dallas, 553–567.

Yusuke Takamatsu and Kenji Kono. 2014. Clickjuggler: Checking for Incomplete Defenses Against Clickjacking. In *Proceedings of the Twelfth Annual International Conference on Privacy, Security and Trust (PST 2014)*. IEEE, Toronto, 224–231.

Gareth Tyson, Shan Huang, Felix Cuadrado, Ignacio Castro, Vasile C. Perta, Arjuna Sathiaseelan, and Steve Uhlig. 2017. Exploring HTTP Header Manipulation In-The-Wild. In *Proceedings of the 26th International Conference on World Wide Web (WWW 2017)*. ACM, Perth, 451–458.

Marie Vasek, John Wadleigh, and Tyler Moore. 2016. Hacking Is Not Random: A Case-Control Study on Webserver-Compromise Risk. *IEEE Transactions on Dependable and Secure Computing* 13, 2 (2016), 206–219.

W3C. 2016. Content Security Policy Level 3, W3C Working Draft. (2016). World Wide Web Consortium (W3C). Available online in January 2018: https://www.w3.org/TR/CSP/.

W3C. 2017. Referrer Policy, W3C Candidate Recommendation. (2017). World Wide Web Consortium (W3C). Available online in January 2018: https://www.w3.org/TR/referrer-policy/.

Joel Weinberger, Prateek Saxena, Devdatta Akhawe, Matthew Finifter, Richard Shin, and Dawn Song. 2011. A Systematic Analysis of XSS Sanitization in Web Application Frameworks. In *Proceedings of the 16th European Symposium on Research in Computer Security (ESORICS 2011), Lecture Notes in Computer Science (Volume 6879)*, Vijay Atluri and Claudia Diaz (Eds.). Springer, Leuven, 150–171.

Michael Weissbacher, Tobias Lauinger, and William Robertson. 2014. Why Is CSP Failing? Trends and Challenges in CSP Adoption. In *Proceedings of the 17th International Symposium on Research in Attacks, Intrusions and Defenses (RAID 2014), Lecture Notes in Computer Science (Volume 8688)*, Angelos Stavrou, Herbert Bos, and Georgios Portokalidis (Eds.). Springer, Gothenburg, 212–233.

Ming Ying and Shu Qin Li. 2016. CSP Adoption: Current Status and Future Prospects. *Security and Communication Networks* 9, 17 (2016), 4557–4573.

Alain F. Zuur, Elena N. Ieno, and Graham M. Smith. 2007. *Analysing Ecological Data*. Springer, New York.