Mobile Friendly or Attacker Friendly? A Large-scale Security Evaluation of Mobile-first Websites

Tom Van Goethem[*]
imec-DistriNet, KU Leuven
tom.vangoethem@cs.kuleuven.be

Victor Le Pochat[*]
imec-DistriNet, KU Leuven
victor.lepochat@cs.kuleuven.be

Wouter Joosen
imec-DistriNet, KU Leuven
wouter.joosen@cs.kuleuven.be

ABSTRACT

In the last few years, traffic generated by mobile devices has surpassed desktop visits. In order to provide users with the best browsing experience, many website owners specifically tailor their site to mobile devices. While some websites make use of reactive designs, many others opt to create an entirely new "mobile-first" website, typically hosted on a subdomain of the desktop site. These mobile-first sites provide a unique viewpoint on how organizations handle security: the mobile version of a site is typically developed several years after the desktop site by the same organization. Through a large-scale security analysis on 10,222 domains with both a desktop and mobile-first version, we find several strong indicators that security is generally applied consistently across the different parts of an organization's web estate. Overall, we find relatively few differences between the desktop and mobile versions of a website, both on the adoption and the implementation of security features, indicating that these are applied reactively rather than proactively during the design phase.

ACM Reference Format:
Tom Van Goethem, Victor Le Pochat, and Wouter Joosen. 2019. Mobile Friendly or Attacker Friendly? A Large-scale Security Evaluation of Mobile-first Websites. In *ACM Asia Conference on Computer and Communications Security (AsiaCCS '19), July 9–12, 2019, Auckland, New Zealand*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3321705.3329855

1 INTRODUCTION

With web traffic generated by mobile devices surpassing desktop traffic in the last two years [23], many organizations have made extensive efforts to create an equivalent browsing experience for mobile users. Typically, this is done by adapting the existing website with a reactive design such that the same site can be used to serve mobile and desktop clients. Another option is to create a mobile-first site, which is specifically tailored to the needs of mobile users, and is often hosted on a subdomain, in conjunction with the site that serves desktop users.

In this paper, we perform a comparative evaluation of mobile-first sites and their desktop counterpart, driven by a large-scale statistical analysis. The unique characteristics of mobile-first sites allow us to make more general observations on how security is applied on the web. More precisely, mobile-first sites are developed by and/or for the same organization as the desktop version, but at a much later time than the desktop version; by analyzing the similarities and differences, we obtain insights on whether security features are typically retroactively applied in an ad-hoc manner, or whether they are the result of a more structured and thorough approach during the initial development phase.

Through a large-scale statistical analysis of the adoption of a wide variety of security features on 10,222 websites, we find that desktop sites exhibit a slightly larger prevalence of these features, both for the positive and negative ones. We validate that this is in part due to the higher complexity of desktop sites. Moreover, we find that almost all security features are consistently adopted on both desktop and mobile versions. For instance, of the 502 desktop sites that leverage Content Security Policy to improve security for their visitors, we find that 83.27% also implement this feature on their mobile site. This, along with the other findings in our paper, are strong indicators that security is applied holistically, across the various aspects of an organization's web estate. Moreover, the minimal differences between desktop and mobile versions, where the latter are typically developed several years later, indicate that security measures are typically applied retroactively instead of during the design or initial development phase.

Next to our statistical approach, we also perform an in-depth analysis of how certain security features are implemented. More specifically, we analyze the adoption of different CSP directives and the deployment of HTTPS. This analysis shows that even in the configuration of security features, there is little difference between mobile and desktop sites, confirming our earlier intuition that security is applied consistently across the various components of a website.

In summary, we make the following contributions:

- We present the first comprehensive analysis on mobile-first sites based on a variety of security features, measured across 10,222 domains visited with both a desktop browser and an emulated mobile browser.
- Leveraging the unique viewpoint that mobile-first websites provide on an organization's general security posture, we determine that security is generally applied in a holistic manner, i.e. consistently across the different web-facing assets of an organization.
- As a result of our in-depth analysis on specific security features, we discover that many sites, both desktop and mobile, are left unprotected despite their developers' intentions.

[*]The authors contributed equally to this paper.

2 DATASET

2.1 Dataset composition

In our evaluation, we set out to explore whether mobile-first websites provide better, or at least on-par, security for their visitors. In order to do so, we first compose a set of mobile-first websites, i.e. websites that are specifically designed to serve mobile users. A common technique that is used to provide a mobile-first website is to host it on a subdomain, such as `m.example.com`, or an entirely different domain, e.g. `example.mobi`. Consequently, when a user navigates to the home page of the website, this website determines whether the user agent is a mobile device by looking at the `User-Agent` request header, or through some JavaScript code (e.g. analyzing the `navigator.userAgent` string or using the `matchMedia` API. If the user agent indeed matches a mobile device, the server will issue a redirect to the mobile-first website.

To determine websites with a (sub-)site specifically designed for mobile devices, we visited the homepage of the Tranco top one million list [15], created on December 20th, 2018[1]. To ensure that our crawler would be redirected to the mobile-first site, we emulated a mobile environment in our headless Chrome (v71) browser: we set the `User-Agent` string to that of Chrome on a popular mobile device, we overrode the device metrics to match those of the mobile device, and used the built-in mobile emulator [6]. For each of the 1 million websites, we visited it once with the mobile browser emulation enabled, and once with the emulation disabled, totaling in two million page visits. For every visit, we recorded the redirection chain, and only considered a website to have a mobile-first component when the "mobile browser" was redirected to a different domain or subdomain than the "desktop browser".

Out of the 15,541 domains that redirected to a different (sub)-domain for mobile, we excluded 512 domains that we found to be compromised. More specifically, when the domain was visited with a desktop browser, the legitimate website was returned, however on the mobile browser, the user was redirected to a supposedly malicious domain, in most cases `wwwtype.ru`. This domain would then again redirect the user to another advertising-based domain, and ultimately the user would end on `m.chaturbate.com` or a website serving suspicious-looking advertisements or fake surveys. The sites were labeled as compromised by means of manual analysis, guided by a targeted search. By obtaining part of the contents of a compromised website, we determined that the adversary added an `.htaccess` file that would match the User-Agent string of mobile browsers and only redirect those to the malicious domain. Presumably, this cloaking tactic is used to extend the lifetime of the compromise as administrators of websites that are typically only contacted on desktop may not immediately notice the malicious behavior. Furthermore, the compromise may be overlooked by search engines and security scanners, if these do not use a mobile browser to crawl the website.

Next, we applied several other filtering steps: we excluded 45 domains that redirected to the Google Play store, and thus do not provide a mobile-first site. Eight domains that only served malware were also excluded from our dataset. Next, we filtered out 268 websites because either the mobile or desktop site was not accessible. Finally, 2,173 sites were excluded because no same-site links could be found on the home page on either the desktop or mobile version. These were mainly pages resulting in an error or redirecting to a third party to ask for cookie consent following the GDPR regulation.

In a next step, we aimed to remove any bias towards websites that are overrepresented in the ranking. For instance, several website have multiple domains with a different TLD, for instance `google.de` and `google.com`. As these refer to the same website, only a single instance of these should be used. To find website from the same owner hosting the same content, we take a perceptual hash [30] of the screenshot of the website's homepage, and cluster these according to their similarity. Next, we manually analyze the different clusters and remove entries that were incorrectly marked as part of a group (our similarity threshold was set high enough to ensure dynamic websites would also be detected as similar). For each cluster of similar domains, we only keep a single one, namely the one that was ranked highest in the Tranco list. In total, 820 domains were removed as duplicates of another domain in the list.

In the same vein, we excluded websites from our dataset that on the mobile version would redirect to the same domain as one of the other websites would on the desktop version, and similarly for desktop sites redirecting to a domain encountered elsewhere. This filtering phase accounted for 1,471 sites that were excluded from our dataset. As such, a total of 5,297 sites were removed from our initial dataset because these do not represent unique websites built specifically for mobile users. In the remainder of this paper, we evaluate the security of 10,222 mobile-first websites that are part of our dataset.

Although in total, we only discovered 10,222 mobile-first websites, i.e. roughly one percent of our starting set of one million, it should be noted that because of our strict criteria, we may have missed certain websites, for instance if a website would host its mobile version on the same fully-qualified domain name (FQDN) as the desktop version, e.g. by setting a cookie that determines which version of the site would be shown. Nevertheless, we believe that our filtering approach did not bias the dataset towards the adoption of security features, and therefore we consider the dataset to be representable of websites with a mobile-first site.

2.2 Mobile site age

Finally, we explore when mobile sites are typically created, in comparison to the desktop version. For this, we leverage the Mementos [26] accessed through the Time Travel service[2], which can be used to discover entries in a variety of web archival servers. We use this service to determine the first occurrence of the mobile (sub)domain and compare it to when the desktop (sub)domain was first encountered. In Figure 1, we show the cumulative distribution function of the time difference in years between the first encounter of the two site versions over the 8,459 domains for which we could determine the age for both the desktop and mobile version. While there are certain limitations to this approach (e.g. the mobile-first version might not be as popular and therefore is less likely show up close to its inception; on the other hand, the desktop version may have existed before the archival services took note of it, making

[1]https://tranco-list.eu/list/RJ2Y/1000000

[2]http://timetravel.mementoweb.org/

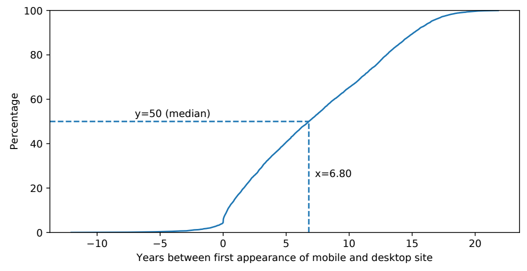

Figure 1: Difference in the first appearance of the mobile site in comparison to the desktop version.

the difference higher than reported), we find that mobile sites are generally developed much later than desktop sites (median: 6.8 years). This provides a unique viewpoint on how security is applied throughout the web: in case security countermeasures are mainly or only considered during the design time, we would observe a significant difference in the adoption of certain security features that were not yet available at the time that the desktop site was developed, but that were available and supported by all major browsers when the mobile site was designed and developed. Conversely, a lack of differences in the adoption of security measures would indicate that these are applied retroactively and consistently across all web-facing assets as an independent effort. Our large-scale statistical analysis presented hereafter, strongly hints towards the latter.

3 EVALUATION OF SECURITY INDICATORS

In this paper, we analyze the security of mobile-first websites. Ideally, we would measure the "absolute" security, i.e. determining whether websites do not suffer from any vulnerabilities, however, this would be infeasible for several reasons. For instance, actively looking for vulnerabilities would be both unethical and illegal. Furthermore, as websites can only be accessed in a black-box manner and no universal vulnerability detection exists, it would be impossible to determine the presence or absence of vulnerabilities. Instead, we analyze the security of mobile-first sites indirectly: by evaluating the prevalence of certain security features, which are considered best practice and can be used to hinder or completely mitigate attacks, we aim to capture a latent factor of security effort. More precisely, website administrators that adopted a certain security feature have made the conscious decision to do so, and therefore are interested in keeping visitors more secure. Furthermore, these features can be passively observed by visiting the website with an instrumented browser. A list of the various security features we considered in our study can be found in Appendix A.

For every website in our dataset with a mobile-first version, we visited up to 20 web pages, both with an emulated mobile and a desktop browser (the same one used to compose our dataset, as described in Section 2.1). For every visit, we captured all security-relevant information, such as response headers, cookies that were set or scripts and iframes that were included. In total, we collected information from 191,237 web pages visited with a mobile browser (on average, 18.67 pages per site), and 195,487 pages visited with a desktop browser (19.08 pages per site on average). Considering that it is advisable for most security features that they are applied throughout the entire website, we believe that this captures a representable view on a website's security efforts.

3.1 Statistical methods

We examine whether the desktop and mobile websites show different web security properties by determining whether the prevalence distributions of these properties are significantly different. As we measure properties for the desktop and mobile version of the same root domain, we apply statistical tests that are appropriate for paired samples. As we might have visited a different number of pages for a domain, we calculate the proportion of pages that exhibit a security property, in order to assess whether they are applied consistently across a website.

As a baseline test for the significance of paired difference between desktop and mobile sites, we use the Wilcoxon signed-rank test [29], which is non-parametric, i.e. it does not rely on a priori assumptions on the distribution of the data, as opposed to the paired t test. Our null hypothesis is that desktop and mobile domains have the same security properties, i.e. that the paired samples come from populations with the same distribution. The test outputs a two-sided p-value that indicates whether this hypothesis can be rejected with statistical significance or not (lower values of p equal higher significance). It also outputs a statistic from which we calculate the rank-biserial correlation using Kerby's simple difference formula [14]. This metric serves as the effect size by denoting how much the device and the security property are correlated, with a higher value indicating a stronger relationship.

However, our insights into the data as well as previous findings [2] lead us to consider whether differences in the measured security properties can be solely attributed to the differences between the security effort made for the desktop or mobile domain, or whether another factor indirectly affects our measurements. More specifically, we assess whether the complexity of a website may influence the perceived security, as websites that support more features require more effort to be configured as securely as simpler sites. We quantify this complexity based on the definition of our security properties, that is the feature for which we calculate the proportion that is (in)secure. For example, the proportion of cookies that have the `Secure` flag may depend on how many cookies a site sets, as more cookies implies more effort to ensure that every single cookie has the appropriate flag.

To determine whether this complexity has an indirect effect on the measured security properties and therefore on the (difference of) distribution per device and ultimately our conclusions, we perform a mediation analysis. This analysis tests whether there is a *mediator* variable (here the complexity) that is influenced by the input variable (the device on which the site is visited) and that causes an indirect effect on the outcome variable (the proportion that is (in)secure). We follow the approach of Montoya and Hayes [18], which is applicable to our case of paired samples. The regression model of Montoya and Hayes results in a confidence interval for the indirect effect: if zero does not lie within this confidence interval, the presence of this effect is statistically significant. The confidence interval is determined through bootstrapping (repeating the regression with samples drawn from the data with replacement), which does not require the assumption of normality.

The mediation model also computes point estimates of the total, direct and indirect effect of our input variable on the outcome through an ordinary least squares regression; here this linear regression model does imply that the differences between desktop and mobile sites are assumed to be normally distributed. The total effect is the sum of the direct and indirect effect; all three effects can then be interpreted as the number of units of change between desktop and mobile, i.e. the difference in the proportion of (in)securely configured features.

3.2 Results

In this section, we interpret the results of our statistical analysis and report on the most significant and interesting findings; Appendix B lists the full numerical results of this analysis.

Overall, security features are seen more often on desktop than on mobile websites. We see that the effect of the device is most outspoken for the features related to man-in-the-middle attacks, of which several show a relatively large and statistically significant skew in adoption toward desktop sites. In general, pages are more likely to be served over HTTPS on the desktop; in section 4.2, we elaborate on the adoption and (in)secure configuration of HTTPS across desktop and mobile websites.

The `Referrer-Policy` header has the highest (and a significant) correlation at 0.429, being more prevalent on the desktop. For the other features and corresponding vulnerabilities, the effect of the device is at best very moderate and usually statistically insignificant, indicating a more consistent application of security features across desktop and mobile sites.

Our mediation analysis shows that the complexity of a website has a significant indirect effect on the proportion of securely configured cookies and frames as well as pages served over HTTPS. Desktop sites tend to have more cookies, and it tends to be less common for sites with more cookies to have a higher proportion with the `HTTPOnly` or `Secure` attribute. The reduction in effect caused by this increased complexity therefore actually counteracts the larger direct effect where desktop sites are even more likely to have more securely configured cookies.

For frames, the same reasoning on complexity holds as desktop sites have more frames which leads to a lower prevalence of the `sandbox` attribute, but here this actually reinforces the finding that mobile sites tend to have more frames with the `sandbox` attribute.

When we analyze the distribution of sites in terms of the number of security properties for which the site has at least one instance, we see a similar picture: Figure 2 shows that desktop sites tend to have implemented more features that positively affect security. However, our analysis paints a bleak picture of security for both desktop and mobile sites: only 530 (5.18%) desktop and 415 (4.06%) mobile sites have implemented at least half of the measured vulnerability mitigations on at least one page, with only 327 (3.20%) domains having done so on both. 6,128 (59.95%) domains see the same number of positive features between the desktop and mobile sites, again with a skew toward both sites having less features rather than more.

Figure 3 shows that the share of desktop versions of sites that exhibit a security property is consistently higher than that of the mobile version. The most widely implemented feature is serving the site over HTTPS, at 6,904 (67.54%) desktop and 6,649 (65.05%) mobile sites with at least one HTTPS page. However, except for the `HttpOnly` cookie attribute, positive security features are present on less than 2,500 domains, with the majority having the feature on both desktop and mobile. Conversely, negative features are affected less by the website type. This shows that when website operators implement good practices, they tend to do this on both desktop and mobile, but that they are less successful at refraining from bad practices in a consistent manner.

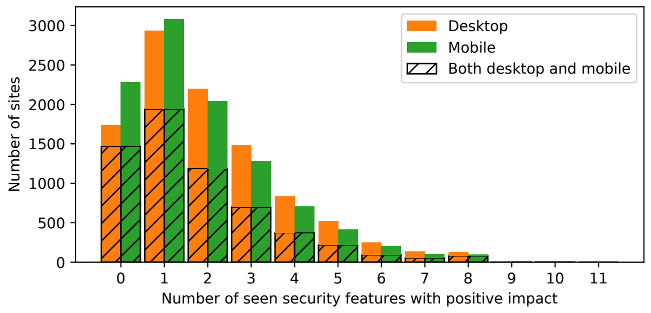

Figure 2: Distribution of sites over the number of properties for which a site has at least one securely configured page.

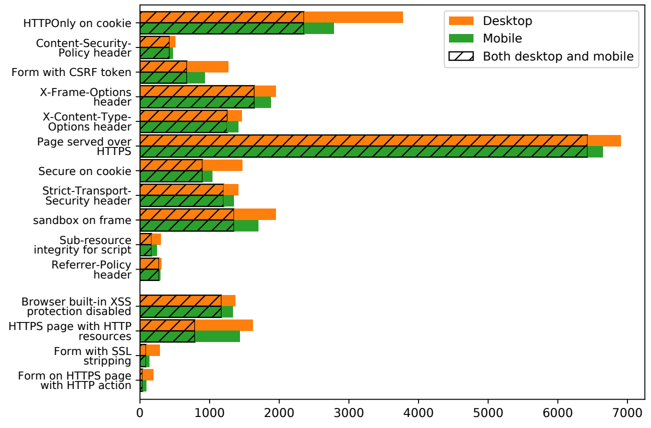

Figure 3: Number of sites with at least one page that exhibits a positive (top) or negative (bottom) feature.

4 IN-DEPTH ANALYSIS OF SECURITY PROPERTIES

In Section 3, we primarily focused on the broad view of how security mechanisms are applied on mobile websites compared to their desktop version. In this section, we take a more detailed look at a few security aspects, which provides more insights on how website operators aim to improve their security, and which kind of challenges they are still facing.

4.1 Content Security Policy

In our dataset, we find 463 mobile-first sites with CSP, and 502 desktop sites that use the feature. Interestingly, having a CSP on the desktop website is a strong indicator that it will also be present on the mobile version: 418 (83.27%) of the CSP-enabled desktop sites also define a version on their mobile site. This indicates that security

Table 1: Percentage of unique CSP policies that use a specific directive, based on data from this study (mobile and desktop) and the study by Weichselbaum et al. [28].

Directive	mobile %	desktop %	[28] %
`report-uri`	41.27	42.60	41.42
`default-src`	34.62	33.94	85.71
`block-all-mixed-content`	33.73	27.08	1.20
`script-src`	29.76	25.99	86.78
`frame-ancestors`	28.17	22.74	8.12
`referrer`	27.78	25.27	1.61
`img-src`	26.59	25.99	77.58
`style-src`	23.41	22.02	78.22
`font-src`	19.84	16.62	66.55
`connect-src`	19.44	20.22	54.37

efforts are typically made universally, across all web-facing assets of an organization. This is confirmed by the fact that on average the percentage of web pages within a site that are protected with CSP is quite high (82.01% for mobile, 78.94% for desktop). Overall, we find that websites that adopt CSP typically create one universal policy that is applied throughout the website: 92.22% and 91.24% of the websites only use a single policy on their mobile and desktop version respectively.

To further evaluate the CSP policies, we use Google's CSP Evaluator[3], which is based on the results of a large-scale study [28]. Similar to the findings of this study from 2016, we find that the vast majority of CSP policies can be bypassed: only 3 mobile-first sites do not suffer from high-severity issues that make the policy ineffective against XSS. On desktop, there are only 2 websites with an effective policy. Similar to the two large-scale studies on CSP that were performed in 2016 [8, 28], we find that most policies are still rendered ineffective due to the use of `'unsafe-inline'` (92.38% on mobile, 94.17% on desktop).

Interestingly, when comparing the prevalence of CSP directives of mobile and desktop sites from our dataset with the findings of Weichselbaum et al. [28], as listed in Table 1, we find significant differences. In general, we observe that there is a much higher variety of CSP directives. For instance, we see that the `block-all-mixed-content`, `frame-ancestors`, and `referrer` directives are significantly more prevalent in our dataset than in the 2016 study. Finally, we can see that there are relatively few differences in the adoption of CSP directives between mobile and desktop sites.

4.2 HTTPS adoption

Figure 4 shows how the desktop and mobile versions of a site compare in terms of secure HTTPS implementations. For 4,473 (43.76%) domains, both are fully secure. In addition, 665 (6.51%) desktop and 386 (3.78%) mobile sites are fully secure while their counterpart is not: for mobile, this is mostly due to the inclusion of HTTP resources on the desktop site, while for the desktop, this is rather due to the mobile site redirecting to HTTP. In fact, redirections to HTTP pages represent the second largest class for mobile sites: for 1,637 (16.01%) sites, a user that visits the root domain over HTTPS is redirected to the mobile version served over HTTP, undoing the additional security of a HTTPS connection.

[3]https://csp-evaluator.withgoogle.com/

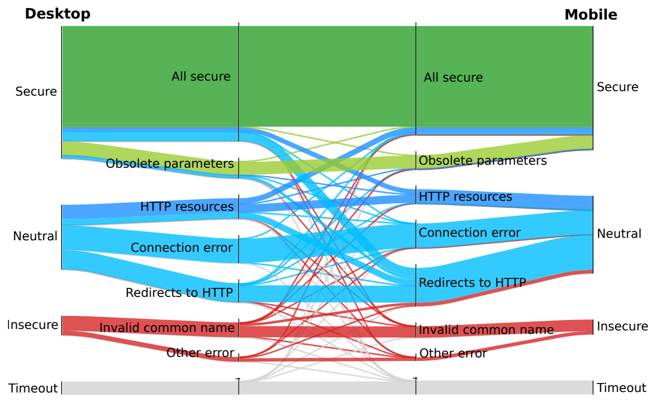

Figure 4: Sankey diagram of the distribution of sites that (in)correctly implement HTTPS on desktop (left) and mobile (right). Flows indicate whether the desktop and mobile site for a domain are implemented as (in)securely; the color represents the least secure configuration of the pair.

650 (6.36%) domains have both an insecure desktop and mobile site, mostly due to the certificate having an invalid common name. 38 insecure desktop and 9 insecure mobile sites have a fully secure site for the other device, indicating that the website operator has only considered one device when setting up HTTPS for their domain. Finally, configurations with obsolete parameters (e.g. an older TLS version) or failures to connect over HTTPS appear to occur consistently between desktop and mobile sites. Overall, less than half of our mobile-first sites are configured securely, on both desktop and mobile; we see that website operators are prone to forget that both versions need to be set up properly, e.g. by redirecting to a HTTP site or having an insecure configuration for one device, leaving some users vulnerable to man-in-the-middle attacks.

5 RELATED WORK

Differences between the desktop and mobile versions of websites in terms of security characteristics can lead to abuse going undetected or the introduction of additional vulnerabilities. Amrutkar et al. [2] demonstrated that static features used in detecting malicious webpages, such as the number of scripts or frames, are less prevalent on mobile than on desktop websites. Therefore, they developed a static analyzer that detects malicious mobile webpages using a tailored set of features. They evaluated their tool on 53,638 mobile webpages found by applying heuristics (subdomains, TLDs and URL paths) that indicate a mobile webpage. Mendoza et al. [17] analyzed inconsistencies in security-related HTTP headers when websites are served to desktop or mobile clients, finding over 2,000 websites with at least one different configuration. They attribute these errors in part to the difficulty of consistently maintaining multiple versions of one website.

Large-scale analyses of web security mechanisms have been used to determine the security awareness and effort of website infrastructure operators. Van Goethem et al. [27] measured defensive security mechanisms and vulnerabilities across over 22,000 European websites, based upon which they develop a score that estimates the security level of a website. Stock et al. [24] performed a longitudinal study of the adoption of security mechanisms, translating them into security awareness indicators, and correlated these with the presence of related vulnerabilities. Tajalizadehkhoob et al. [25] measured security features on shared hosting platforms, and derived the security effort of both the website operator and hosting provider through statistical inference of latent factors.

Due to the particular environment, e.g. a small screen size, browsing on mobile devices can suffer exploits that are not present on the desktop. Niu et al. [20] were the first to uncover additional vectors for phishing across three mobile browsers, e.g. due to URL truncation. Felt and Wagner [9] found that the interaction between mobile applications and websites visited through mobile browsers can also enable phishing attacks. Amrutkar et al. [3] found two classes of display security vulnerabilities in mobile browsers that were previously unseen in desktop versions. The authors later also analyzed the visibility of security indicators in mobile browsers [4]. Luo et al. [16] studied support for security mechanisms in mobile browsers over time, finding that while adoption increases over time, several popular mobile browsers do not yet support a majority of such mechanisms, and may be slow to pick up features already in use on popular websites.

6 CONCLUSION

In this paper, we set out to explore whether the security of mobile-first sites is comparable to that of their desktop counterpart. Through a large-scale analysis of a variety of security features on 10,222 websites, we find that in most cases desktop sites exhibit a minimally higher prevalence and wider-reaching coverage of these features compared to mobile sites. As mobile sites are typically developed by the same organization, but several years after the desktop version, the lack of difference in the adoption and coverage of security features and the similarities in their implementation strongly suggest that for many organizations, security is a retroactive effort that is applied consistently across all assets.

ACKNOWLEDGMENTS

This research is partially funded by the Research Fund KU Leuven. Victor Le Pochat holds a PhD Fellowship of the Research Foundation - Flanders (FWO).

REFERENCES

[1] Maarten Aertsen, Maciej Korczyński, Giovane Moura, Samaneh Tajalizadehkhoob, and Jan van den Berg. 2017. No domain left behind: is Let's Encrypt democratizing encryption?. In *Proceedings of the Applied Networking Research Workshop (ANRW 2017)*. ACM, 48–54.

[2] C. Amrutkar, Y. S. Kim, and P. Traynor. 2017. Detecting Mobile Malicious Webpages in Real Time. *IEEE Transactions on Mobile Computing* 16, 8 (Aug 2017), 2184–2197.

[3] Chaitrali Amrutkar, Kapil Singh, Arunabh Verma, and Patrick Traynor. 2012. VulnerableMe: Measuring Systemic Weaknesses in Mobile Browser Security. In *Proceedings of the 8th International Conference on Information Systems Security (ICISS 2012)*. Springer, 16–34.

[4] C. Amrutkar, P. Traynor, and P. C. van Oorschot. 2015. An Empirical Evaluation of Security Indicators in Mobile Web Browsers. *IEEE Transactions on Mobile Computing* 14, 5 (May 2015), 889–903.

[5] Sajjad Arshad, Seyed Ali Mirheidari, Tobias Lauinger, Bruno Crispo, Engin Kirda, and William Robertson. 2018. Large-Scale Analysis of Style Injection by Relative Path Overwrite. In *Proceedings of the 2018 World Wide Web Conference (WWW 2018)*. International World Wide Web Conferences Steering Committee, 237–246.

[6] Kayce Basques. 2018. Simulate Mobile Devices with Device Mode in Chrome DevTools. https://developers.google.com/web/tools/chrome-devtools/device-mode/.

[7] Frederik Braun and Mario Heiderich. 2013. X-Frame-Options: All about Clickjacking? "How else do X-Frame-Options protect my website". https://cure53.de/xfo-clickjacking.pdf. (2013).

[8] Stefano Calzavara, Alvise Rabitti, and Michele Bugliesi. 2016. Content security problems?: Evaluating the effectiveness of content security policy in the wild. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*. ACM, 1365–1375.

[9] Adrienne Porter Felt and David Wagner. 2011. Phishing on Mobile Devices. In *Web 2.0 Security and Privacy 2011 (W2SP 2011)*.

[10] Gertjan Franken, Tom Van Goethem, and Wouter Joosen. 2018. Who left open the cookie jar? A comprehensive evaluation of third-party cookie policies. In *Proceedings of the 27th USENIX Security Symposium*. 151–168.

[11] Tobias Gondrom and David Ross. 2013. HTTP header field X-Frame-Options. https://tools.ietf.org/html/rfc7034. (2013).

[12] Mario Heiderich, Marcus Niemietz, Felix Schuster, Thorsten Holz, and Jörg Schwenk. 2012. Scriptless attacks: stealing the pie without touching the sill. In *Proceedings of the 2012 ACM conference on Computer and communications security (CCS 2012)*. ACM, 760–771.

[13] Gareth Heyes. 2014. RPO. http://www.thespanner.co.uk/2014/03/21/rpo/.

[14] Dave S. Kerby. 2014. The Simple Difference Formula: An Approach to Teaching Nonparametric Correlation. *Comprehensive Psychology* 3 (Jan. 2014), 1:1–1:9.

[15] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczyński, and Wouter Joosen. 2019. Tranco: A Research-Oriented Top Sites Ranking Hardened Against Manipulation. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*.

[16] Meng Luo, Pierre Laperdrix, Nima Honarmand, and Nick Nikiforakis. 2019. Time Does Not Heal All Wounds: A Longitudinal Analysis of Security-Mechanism Support in Mobile Browsers. In *Proceedings of the 26th Annual Network and Distributed System Security Symposium (NDSS 2019)*.

[17] Abner Mendoza, Phakpoom Chinprutthiwong, and Guofei Gu. 2018. Uncovering HTTP Header Inconsistencies and the Impact on Desktop/Mobile Websites. In *Proceedings of the 2018 World Wide Web Conference (WWW 2018)*. ACM.

[18] Amanda K. Montoya and Andrew F. Hayes. 2017. Two-condition within-participant statistical mediation analysis: A path-analytic framework. *Psychological Methods* 22, 1 (2017), 6–27.

[19] Nick Nikiforakis, Luca Invernizzi, Alexandros Kapravelos, Steven Van Acker, Wouter Joosen, Christopher Kruegel, Frank Piessens, and Giovanni Vigna. 2012. You are what you include: large-scale evaluation of remote javascript inclusions. In *Proceedings of the 2012 ACM conference on Computer and communications security (CCS 2012)*. ACM, 736–747.

[20] Yuan Niu, Francis Hsu, and Hao Chen. 2008. iPhish: Phishing Vulnerabilities on Consumer Electronics. In *Proceedings of the 1st Conference on Usability, Psychology, and Security (UPSEC 2008)*.

[21] Gustav Rydstedt, Baptiste Gourdin, Elie Bursztein, and Dan Boneh. 2010. Framing Attacks on Smart Phones and Dumb Routers: Tap-jacking and Geo-localization. In *Proceedings of the 4th USENIX Workshop on Offensive Technologies*.

[22] Emily Schechter. 2018. A milestone for Chrome security: marking HTTP as "not secure". https://www.blog.google/products/chrome/milestone-chrome-security-marking-http-not-secure/.

[23] Statcounter. 2019. Desktop vs Mobile Market Share Worldwide - Jan 2014 - Dec 2018. http://gs.statcounter.com/platform-market-share/desktop-mobile/worldwide/#monthly-201401-201812.

[24] Ben Stock, Martin Johns, Marius Steffens, and Michael Backes. 2017. How the Web Tangled Itself: Uncovering the History of Client-Side Web (In)Security. In *Proceedings of the 26th USENIX Security Symposium (USENIX Security 2017)*. USENIX Association, 971–987.

[25] 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, 553–567.

[26] Herbert Van de Sompel, Michael L Nelson, Robert Sanderson, Lyudmila L Balakireva, Scott Ainsworth, and Harihar Shankar. 2009. Memento: Time travel for the web. *arXiv preprint arXiv:0911.1112* (2009).

[27] Tom Van Goethem, Ping Chen, Nick Nikiforakis, Lieven Desmet, and Wouter Joosen. 2014. Large-scale security analysis of the web: Challenges and findings. In *Proceedings of the 7th International Conference on Trust and Trustworthy Computing (TRUST 2014)*. Springer, 110–126.

[28] Lukas Weichselbaum, Michele Spagnuolo, Sebastian Lekies, and Artur Janc. 2016. CSP is dead, long live CSP! On the insecurity of whitelists and the future of content security policy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security (CCS 2016)*. ACM, 1376–1387.

[29] Frank Wilcoxon. 1945. Individual Comparisons by Ranking Methods. *Biometrics Bulletin* 1, 6 (Dec. 1945), 80–83.

[30] Christoph Zauner. 2010. Implementation and benchmarking of perceptual image hash functions. http://www.phash.org/.

A SECURITY FEATURES

The ever-increasing complexity of the modern-day web has introduced a wide variety of vulnerability classes. In this section, we give a brief summary of the most common ones and discuss some of the mechanisms that are used to mitigate them: these are the security features that we study in this paper.

A.1 Cross-site scripting (XSS)

The vast majority of modern websites are dynamically composed with content originating from many different sources, such as third-party services or information contained in the GET or POST request. A cross-site scripting vulnerability is formed when a web page includes content from an untrusted, attacker-controlled source without sufficient encoding or escaping. Most commonly, the value of a GET or POST query parameter is put directly in the response: when the website developer did not correctly encode this value, e.g. by replacing < with `&lt;`, an attacker could include a malicious script in the query parameter. As a response to the prevalence of this type of attack, several browser vendors have implemented a built-in countermeasure, which is enabled by default. Website owners can control this protection mechanism through the `X-XSS-Protection` response header.

Although it is generally advised to sanitize all untrusted input, several defense-in-depth mechanisms have been developed to completely mitigate XSS attacks or minimize their impact. A well-known instance of the latter is the `HttpOnly` attribute on cookies: browsers prevent these cookies from being read by JavaScript code, effectively safeguarding them from XSS attacks. A more extensive defense mechanism is Content Security Policy (CSP), which can be used by website operators to instruct their visitor's browser which content, e.g. scripts, can be included, thus blocking any malicious scripts.

A.2 Cross-site request forgery (CSRF)

Cross-site request forgery (CSRF) is both very common and can have very severe consequences, such as full account compromise. The vulnerability is caused by improperly validating the authenticity of a form submission. More precisely, attackers can replicate a form that is found on the victim website on their own website, i.e. a form with the same endpoint and parameters whose values are controlled by the attacker. The most commonly advised defense against CSRF vulnerabilities is to include a randomly generated token, often referred to as a nonce, with each request that triggers a change. For web forms, this token is typically included as a hidden parameter, whose value cannot be obtained by the adversary due to the Same-Origin Policy. Upon receiving a request, the website then has to validate the value of this token, and otherwise abort processing the request. As long as all forms are protected with this mechanism, this provides a secure and complete mitigation to CSRF attacks.

Another type of defense mechanism aims to tackle the vulnerability by eliminating one of its prerequisites, namely the fact that the cross-site request needs to be authenticated, i.e. the victim's cookie needs to be attached. This defense can be used by setting the `SameSite` attribute on cookies, which prevents them from being sent in a cross-origin manner. At the time of this writing, same-site cookies are still relatively new, but currently supported by all major desktop browsers and most mobile browsers[4], despite several implementation flaws in certain browsers [10].

A.3 Clickjacking

A web page can include other web pages in an `<iframe>` element, even when these are hosted on a cross-origin domain. This functionality introduced so-called clickjacking attacks, also referred to as UI-redressing attacks. In this type of attack, the adversary creates a malicious web page where they include a target web page that for instance contains a button, which when clicked will perform an action on the victim's behalf. On mobile browsers, tricking the user to click on a certain location may be facilitated by leveraging tap-jacking attacks, where the adversary re-creates a part of the visual browser environment which becomes hidden when scrolling down. This makes the victims believe that they are interacting with the browser application, e.g. to switch tabs, while they are actually clicking an overlayed invisible frame [21].

Originally proposed in 2012, the `X-Frame-Options` response header can be used to deter clickjacking attacks [11]. More specifically, this header can be used by website administrators to overrule the default behavior, and prevent other websites from including the protected web pages. Next to clickjacking, `X-Frame-Options` also provides a range of other attacks that rely on framing a target web page [7]. As such, it should be considered best practice to return this header on all endpoints, except those that are explicitly meant to be included. Recent work by Luo et al. reported that in the Google Chrome browser, the `Allow-From` directive of the `X-Frame-Options` header is not supported [16]. Instead, the browser vendor advises website owners to make use of the `frame-ancestors` directive of Content Security Policy.

A.4 Content-sniffing vulnerabilities

This attack could be mitigated by adding the response header To prevent attacks that exploit the browser's content-sniffing algorithm, the `X-Content-Type-Options` with the value `nosniff` can be used. This XCTO header also prevents relative-path override (RPO) attacks [5, 13], which abuse the fact that some websites include stylesheets from relative paths, i.e. by using `../` in the path. A mismatch between what is considered the endpoint on the client-side versus on the server-side, allows an adversary to inject CSS code under certain conditions. This could be used to perform a wide range of attacks against unwitting victims [12].

A.5 Man-in-the-middle attacks

Many recent initiatives are driving the adoption of TLS on the web, e.g. Let's Encrypt allows website owners to conveniently obtain a certificate for free, resulting in a significantly increased adoption [1], and Google Chrome is marking web pages visited over HTTP as "Not Secure" [22]. However, there are still various things that can go wrong in the presence of a man-in-the-middle adversary who can actively manipulate unencrypted requests and responses. For instance, if an HTTPS web page would include a JavaScript file from an HTTP endpoint, a MitM attacker could still manipulate its contents and thus execute arbitrary JavaScript on the website.

[4]https://caniuse.com/#feat=same-site-cookie-attribute

Table 2: Summary of the results of our statistical analysis on pairs of desktop and mobile sites. Stars indicate statistical significance of the test scores (*: p<0.05; **: p<0.01; ***: p<0.001; ****: p<0.0001). The sign of the (in)direct effect indicates whether it goes in the same (+) or opposite (-) direction as the total effect. A mediator is present if zero lies outside the confidence interval of the indirect effect.

							Mediation analysis						
					Wilcoxon		Total effect		Direct effect		Indirect effect		
Feature		Pairs	Different	Direction	Corr.	p	Size	p	Size	p	Size	Confidence interval	Mediator
Cross-site scripting													
`HTTPOnly` on cookie	(+)	8231	3273	desktop	0.142	0.000 (****)	0.47%	0.055	+1.13%	0.000 (****)	-0.66%	[-0.83, -0.51]	Yes
`Content-Security-Policy` header	(+)	10222	299	desktop	0.013	0.847	0.14%	0.105	+0.11%	0.209	+0.03%	[+0.00, +0.06]	Yes
Browser built-in XSS protection disabled	(-)	10222	725	desktop	0.053	0.220	0.10%	0.486	+0.08%	0.578	+0.02%	[-0.02, +0.06]	No
Cross-site request forgery													
Form with CSRF token	(+)	7697	1195	desktop	0.026	0.440	0.16%	0.422	+0.12%	0.567	+0.04%	[-0.11, +0.22]	No
Clickjacking													
`X-Frame-Options` header	(+)	10222	1146	desktop	0.070	0.041 (*)	0.35%	0.034 (*)	+0.32%	0.054	+0.03%	[-0.02, +0.08]	No
Content-sniffing													
`X-Content-Type-Options` header	(+)	10222	772	desktop	0.055	0.189	0.14%	0.320	+0.12%	0.415	+0.02%	[-0.02, +0.07]	No
Man-in-the-middle attacks													
Page served over HTTPS	(+)	10222	2063	desktop	0.222	0.000 (****)	2.19%	0.000 (****)	+2.11%	0.000 (****)	+0.08%	[+0.01, +0.16]	Yes
`Secure` on cookie	(+)	8231	1312	desktop	0.263	0.000 (****)	0.85%	0.000 (****)	+1.17%	0.000 (****)	-0.32%	[-0.41, -0.24]	Yes
`Strict-Transport-Security` header	(+)	6428	639	desktop	0.139	0.002 (**)	1.09%	0.000 (***)	+1.13%	0.000 (***)	-0.03%	[-0.15, +0.06]	No
HTTPS page with HTTP resources	(-)	6428	1833	mobile	0.031	0.245	0.55%	0.038 (*)	+0.61%	0.020 (*)	-0.07%	[-0.15, +0.01]	No
Form with SSL stripping	(-)	5183	201	desktop	0.203	0.012 (*)	0.25%	0.039 (*)	+0.25%	0.047 (*)	+0.00%	[-0.03, +0.03]	No
Form on HTTPS page with HTTP action	(-)	6428	192	desktop	0.391	0.000 (****)	0.35%	0.001 (***)	+0.38%	0.000 (***)	-0.03%	[-0.07, +0.00]	No
Including untrusted content													
`sandbox` on frame	(+)	6893	1986	mobile	0.080	0.002 (**)	0.82%	0.000 (****)	+0.53%	0.010 (*)	+0.29%	[+0.15, +0.48]	Yes
Sub-resource integrity for script	(+)	10180	374	mobile	0.060	0.312	0.01%	0.360	+0.01%	0.557	+0.00%	[+0.00, +0.01]	Yes
Information leakage													
`Referrer-Policy` header	(+)	10222	120	desktop	0.429	0.000 (****)	0.19%	0.000 (***)	+0.18%	0.000 (***)	+0.02%	[-0.00, +0.04]	No

Furthermore, when not all pages of a website are loaded over HTTPS, an attacker could set up an SSL-stripping attack: the MitM attacker sets up a secure connection with the server, and replaces all references of `https://` with `http://`. As such, the server is led to believe that a secure channel is used, and the victim is led to believe that the server does not support HTTPS. A exemplary instance of this is when the website serves all pages over HTTP, but references the login form with HTTPS, as the login-action would contain the user's password. Nevertheless, the attacker can simply perform an SSL-stripping attack and still obtain the user's password.

There exists several other mechanisms that can further enhance the secure connection between a client and the web server. For instance, the `Strict-Transport-Security` mechanism indicates to the browser that this website should only be contacted over a secure channel for a given time; this eliminates the aforementioned SSL-stripping attacks. Similarly, the `Public-Key-Pinning` response header can be used to indicate that the browser should associate only the specified public keys with the web server. When used incorrectly, this mechanism could effectively lock out users from a website, so it requires a significant effort to set up properly. Finally, to ensure certain cookies are only sent over secure channels, the `Secure` attribute could be set.

A.6 Including untrusted content

One of the aspects that allows the web to thrive is that web pages are allowed to include content from any other location. As websites massively include content hosted or provided by third parties, this brings along a variety of security-related issues. For instance, Nikiforakis et al. showed that in many cases web developers included JavaScript files from a domain that later became available for registration [19]. An adversary could register one of these stale domain names and serve malicious scripts from it, thereby affecting many users. A countermeasure that can be used to minimize the trust that needs to be put in third-party content providers (other than not re-registering their domain name, these could also act maliciously or be compromised), is sub-resource integrity (SRI). The mechanism allows websites to set the `integrity` attribute on `<script>` elements, with the value set to the hash of the expected content of the included script. This script is only executed when the integrity check passes, i.e. the hash of the actual content matches the provided hash value.

B EVALUATION OF SECURITY INDICATORS: STATISTICAL ANALYSIS

In Table 2, we summarize the numerical results of our statistical analysis from Section 3, for the various weaknesses and mitigation techniques that we considered.