Is Website Popularity an Indicator of Better Security?

Luis Ulloa
Stanford University

Tim Chirananthavat
Stanford University

Abstract

In this paper, we investigate the question of whether popular websites are more secure. We use the Tranco Top 1M list as an indicator for website popularity [1]. For an indicator of website security, we use the numerical score provided by completing a Mozilla Observatory scan on the website [2]. We compare the top 10,000 sites to the remaining 990,000 sites and find that the top 10,000 sites average more than double the score of the other 990,000 sites. We inspect which tests the top sites performed better on and found evidence for faster HTTPS adoption by top sites, which is inline with work by Mirian et al. [3]. However, we do not see a very strong correlation between popularity and security when using the Spearman Rank Correlation Coefficient on a random sample of the Tranco list. We look into the significance of the web server hosting each domain and find that the servers with the highest Mozilla Observatory scores were proprietary web servers provided by web hosting solutions, such as Google Web Server.

1 Introduction

Are more popular websites more secure? This is a question that most people would assume has an obvious answer, since when someone thinks "popular website," they think of keystone sites like www.google.com and www.facebook.com which are hosted by prominent technology companies and have been around for years. However, there are billions of websites on the internet and the average person visits dozens of unique domains each month, making it highly unlikely that they are always visiting sites with a dedicated security team.

Having a belief in popular sites being significantly more secure can be very problematic from a security standpoint since the number of people that could be affected by a security vulnerability increases proportionately with a site's popularity. For example, if a popular site was to be involved in a database leak (which we have seen repeatedly in recent years), then sensitive information like a user's email and password could be leaked. Since the user trusted the site's security, it is possible they re-used the same email and password for other sites that they trusted, which could lead to a user's account on secure sites being compromised as well.

To give credence to the idea that popular software is not necessarily more secure software, prior work by Siavvas et al. determined that popularity was not a reliable indicator of security in open source libraries [4]. This result inspired the parallel question for website popularity and security. Website popularity and website security are both unclear metrics that must be defined for this study. For website popularity, we use the Tranco Top 1M list, which is a stable website popularity ranking that takes into account factors like time on site and traffic [1]. For website security, we use the Mozilla Observatory security scanner. This scanner tests for common web security practices and assigns a numeric score based on a website's performance on these tests, which are largely configuration based.

Contributions: Our contributions are summarized as follows:

- A data collection pipeline for efficiently sampling and processing a large number of Mozilla Observatory scans, and a post-processing step for identifying which web server is associated with each domain.
- An initial analysis of the relationship between website popularity and website security, and a look into the feasibility of Mozilla Observatory as an indicator of security.
- Potential evidence for faster HTTPS adoption by more popular sites (see associated discussion for strict-transport-security in Table 1).
- Evidence for proprietary web hosting solutions being more secure than other solutions (see Table 3).

2 Related Work

Siavvas et al. investigated whether popularity was an indicator of software security in open-source software applications and libraries [4]. This was done by first calculating a popularity score based on the number of downloads for the software and a security score generated with Static Analysis Vulnerability Density (SAVD) [5]. Then, given the Spearman's rank correlation coefficient between these two scores, it was found that more popular libraries actually tended to be less secure, though this relationship was weak. The final conclusion was that popularity was not a reliable indicator of software security, which is a surprising result that inspires this parallel research question for website security.

HTTPS/TLS is commonly regarded as a crucial web security feature for providing confidentiality and integrity for traffic between a site and its visitor, making it a good indicator for a more secure site. Mirian et al. analyzes the adoption rate of HTTPS by the most popular websites (top 10,000 by Alexa Top 1M) compared to the "longtail" (remaining 990,000 in list) [3]. This work found that 60% of the most popular websites provided HTTPS connectivity while only 45% of the longtail used HTTPS. In addition, they found that hosting provider usage and cost correlated with higher HTTPS adoption (and therefore better security). Since the use of HTTPS is a good indicator for better security, this gives credence to the idea that popular sites have better security. We may run a similar analysis with TLSv1.3 adoption in the longtail during this study, to see if popularity and security are correlated with adoption.

3 Mozilla Observatory

Mozilla Observatory is a public security scanner that tests a given site against a set of web security tests, and then assigns that site a score based on how many of these tests it passes [2].

3.1 Tests & Scoring Methodology

Mozilla Observatory adopts a negative scoring model that begins at 100 and floors at 0. Tests may award bonus points, with the current set of tests allowing for up to 35 total bonus points, making the maximum possible score 135. The worst possible site performance would receive a score of -160 (260 points deducted from 100), but the floor is 0, so the minimum score is 0. In Figure 1, we see that the vast majority of sites perform poorly with respect to Mozilla Observatory's security score.

Mozilla Observatory currently has 12 distinct tests. Below, we enumerate each test and explain what it does and how it is graded.

- **Content security policy** (score range -25 to 10): This is an HTTP header used to prevent XSS attacks. This header specifies how resources (e.g. JavaScript files) are loaded. Up to 10 bonus points are awarded when this CSP header includes `default-src 'none'` (meaning that resources are disallowed from loading unless explicitly listed), and doesn't use `unsafe-inline` (which allows running `<script>` tags without matching hashes in the CSP header) or `unsafe-eval` (which allows calling `eval()` and similar functions on strings without matching hashes in the CSP header).

- **Cookies** (score range -40 to 5): Cookies should be configured to have their access as limited as possible, in order to minimize potential damage from XSS attacks. Up to 5 bonus points are awarded when all cookies use the `Secure` flag, session cookies use the `HttpOnly` flag, and and cross-origin restrictions are in place via the `SameSite` flag

- **Contribute.json** (score range -10 to 0): Contribute.json is a common standard for describing Mozilla websites and projects and is meant to make open source contribution information easier to access. There are no bonus points for no usage or correct usage, but it is possible to *lose* 10 points if contribute.json is implemented incorrectly.

- **Cross-origin resource sharing** (score range -50 to 0): This refers to correctly using the Access-Control-Allow-Origin HTTP header to restrict foreign origin access to content on the site. No bonus points are awarded for implementing this or using it reasonably (e.g. making content not visible), but it is possible to lose up to 50 points if this header is used to give universal access to foreign origins (i.e. no restrictions).

- **HTTP public key pinning** (score range -5 to 0): This refers to using the Public-Key-Pins-Report-Only HTTP header to pin sites to a specific certificate authority or public key, in an effort to prevent compromised certificate authorities from issuing unauthorized certificates. This header is deprecated now in favor of Certificate Transparency, but the test still exists. No bonus points available and any form of usage (or no usage) is acceptable. Including the header in an unrecognizable format leads to points lost.

- **HTTP strict transport security** (score range -20 to 5): The HTTP Strict Transport Security (HSTS) HTTP header notifies the user agent that a site should only be connected to via HTTPS, which allows for browsers to reload the site as HTTPS even if the connection was initiated as HTTP. Up to 5 bonus points are awarded if HSTS is pre-loaded, no points are deducted if the HSTS header has been set for over 6 months, only 10 points are lost if the header has not been set for 6 months yet, and 20 points are lost for sites that do not support HTTPS or have invalid HSTS headers.

- **Redirection** (score range -20 to 0): This test refers to how the site handle redirects that originate from a re-

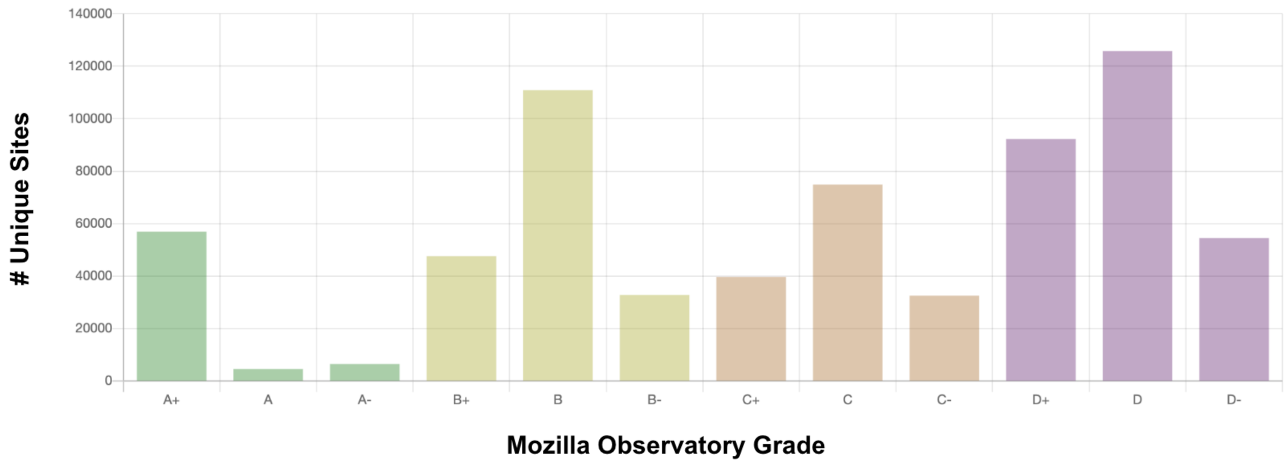

Figure 1: Number of Unique Sites per Grade

source on the site. Regardless of whether the original resource is loaded from the HTTP version of the site, the redirected resource should be on the HTTPS version of the site (or the site should preload HTTPS with HSTS). Up to 20 points can be lost, with redirecting to an HTTP site losing maximum points.

- **Referrer policy** (score range -5 to 5): The Referrer-Policy HTTP header specifies when the Referrer HTTP header can be sent. The Referrer header informs the destination site of the origin of a received request (e.g. what site the user was on when they clicked a link to another site). Referrer poses a privacy concern since it leaks one step of the user's browsing history. Up to 5 bonus points are awarded for having a very restrictive Referrer-Policy, no points are taken away for not using a policy, and points are lost for policies that are unrecognizable or too broad (e.g. `unsafe-url`).

- **Subresource integrity** (score range -50 to 5): Subresource integrity (SRI) ensures that the contents of a JavaScript library match the contents of the library at a particular point in time. Without this, if an attacker modified a third party library and this library was imported by multiple sites, all of those sites would be compromised. This test awards up to 5 bonus points for SRI policies that ensure scripts are loaded securely and/or from safe origins. No points are deducted for cases where SRI is deemed not necessary. Up to 50 points are deducted for not implementing SRI or loading external scripts over HTTP.

- **X-Content-Type-Options** (score range -5 to 0): The X-Content-Type-Options HTTP header is used to ensure that a site does not load a resource that has the incorrect MIME type, since this could lead to XSS attacks where an adversarial non-script file is marked as a script. Passing this test entails setting this header to `nosniff`. Otherwise, 5 points are deducted.

- **X-Frame-Options** (score range -20 to 5): The X-Frame-Options HTTP header allows a site to control how other sites are able to interact with this site in an iframe. Being restrictive helps prevent clickjacking attacks. Up to 5 bonus points are awarded for implementing this using the Content Security Policy (CSP) frame-ancestors directive. Reasonable header values like `ALLOW-FROM uri` and `SAMEORIGIN`/`DENY` pass with no deductions. Not implementing this or having an unrecognizable header is a 20 point deduction.

- **X-XSS-Protection** (score range -10 to 0): The X-XSS-Protection header is used to stop pages from loading when a reflected XSS attack is detected. Any valid configuration passes this test (or using CSP). Disabling this, not implementing it, or having an unrecognizable value leads to a 10 point deduction.

4 Methodology

We first perform data collection using the Mozilla Observatory API, and then we perform an analysis on the collected data. The code for this project can be found here: https://github.com/ulloaluis/cs356-project.

4.1 Data Collection

We perform data collection using the Mozilla Observatory API. The Mozilla Observatory API has three endpoints of interest.

1. Invoke Assessment: Submitting a POST HTTP request to the analyze endpoint with the "host" parameter pro-

vided will begin a Mozilla Observatory scan at the specified "host" domain. The server responds with a Scan object, which contains the last assessment completed on this domain in the past 24 hours (cached entry). If no cached entry, the "state" flag on the Scan object indicates that the scan has begun and is pending.

2. Retrieve Assessment: Submitting a GET HTTP request to the analyze endpoint with the "host" parameter provided will attempt to retrieve an assessment of the last Mozilla Observatory scan completed on the specified "host" domain in the past 24 hours. The assessment is returned as a Scan object that includes higher level aggregation information such as the overall test score and the number of tests passed. If no scan has been completed in the past 24 hours, a scan is still pending, or the last scan failed, the "state" flag is set to indicate this.

3. Retrieve Test Results: Submitting a GET HTTP request to the getScanResults endpoint with the "scan" parameter provided will attempt to retrieve the test results associated with the "scan" scan id. The scan id can be retrieved using either the **Invoke Assessment** or **Retrieve Assessment** endpoint. getScanResults returns a Tests object that includes detailed information about which of Mozilla Observatory's sub-tests failed/passed.

To process a single domain, you must first use the **Invoke Assessment** endpoint to begin the scan. Then, **Retrieve Assessment** and **Retrieve Test Results** can be used to get aggregate score data and per-test data related to the domain once the scan has been completed. If a scan on this domain has already been completed in the past 24 hours, you can retrieve that using only **Invoke Assessment**. The Mozilla Observatory scans take a variable amount of time and are influenced by factors that are mostly out of our control (e.g. overloaded servers). In addition, a scan could fail for a variety of reasons, and these errors could happen at essentially any point: when you invoke an assessment (e.g. invalid domain name) or attempt to retrieve a scan's results (e.g. scan failed due to specified domain being down for maintenance at time of scan).

Given this API, our data collection pipeline is designed to collect data in two phases in order to maximize efficiency. In the first phase, we initiate each scan for the domains we are interested in. In the second phase, we attempt to retrieve the test results. In between these two phases, we typically wait for a preconfigured amount of time depending on the number of domains (around 1-3 minutes is typically fine). When we retrieve the test results, if the scan is still pending, we have additional retry logic in place for robustness, but this is not usually needed.

For our analysis later, we often want to randomly sample N domains from the Tranco list. However, since the scanning process can fail at any point, we often complete the data collection pipeline with only M domains, where $M < N$. In this case, we resample $N - M$ more domains and re-run the data collection pipeline. As an optimization, we always pessimistically over-sample by 20%, under the assumption that 20% of the domains queried will fail to scan for some reason. This reduces the amount of times we have to re-run our data collection pipeline, and it often allows us to collect all N domains in one pass.

Although Mozilla Observatory includes the full response headers associated with their scan, we found that the vast majority of the sites (more than 95%) did not return the Server field to Mozilla Observatory. As a result, we have a post-processing step for the data collected via the pipeline. This post-processing step does a simple HTTP GET request to a domain in order to get the Server header; it then updates our database. In practice, we observed that this reduced the number of missing Server headers from 95% to about 15%, which is a reasonable number for servers that simply do not return the Server header to begin with (likely "security through obscurity").

5 Analysis & Results

We performed three analyses to explore the relationship between website popularity and website security. In our first analysis, we do a brief analysis of a sample of the Tranco Top 1M as a whole, since this gives us some useful aggregate data about websites in general using our chosen security score (Mozilla Observatory). In our second analysis, we compare the most popular sites (top 10,000) to all other sites (remaining 990,000) and attempt to identify the primary differences between these two groups. In our last analysis, we look into the relationship between the web server used by a website and the website's security.

5.1 All Sites

For this analysis, we randomly sampled 1% (10,000) of all sites in the Tranco list. The average score in this sample was 11.57. We then used the Spearman Rank Correlation Coefficient in the Python Scipy library to get a correlation score between Tranco Rank and Mozilla Observatory Score.

Using this metric, we received a correlation coefficient of -0.076 that was very statistically significant (p=2.897e-14). This implies a very weak negative correlation between Tranco Rank and Mozilla Observatory score, which means as the numerical Tranco rank increases (meaning popularity decreases), the score decreases.

5.2 Top Sites vs. Longtail Sites

For this analysis, we randomly sampled 1% (100) from the top 10,000 sites in the Tranco list ("top sites") and 1% (100) from the remaining 990,000 sites ("longtail sites").

Table 1: Per-Test Pass Frequency for Top & Longtail Groups

Test	Top%	Long%	Top Freq	Long Freq
content-security-policy	1.00	0.19	1	19
contribute	100.00	100.00	100	9900
cookies	69.00	67.10	69	6643
cross-origin-resource-sharing	96.00	98.15	96	9717
public-key-pinning	100.00	99.99	100	9899
redirection	48.00	47.93	48	4745
referrer-policy	91.00	97.93	91	9695
strict-transport-security	40.00	14.99	40	1484
subresource-integrity	28.00	35.27	28	3492
x-content-type-options	45.00	23.43	45	2320
x-frame-options	64.00	24.06	64	2382
x-xss-protection	32.00	16.81	32	1664

The average score of the top sites was 23.450 and the average score for the longtail sites was 11.454. This means that top sites averaged more than double the score of longtail sites, which seems like a very significant difference.

One interesting avenue for analysis is to consider how the top sites performed on each test in comparison to the longtail sites, which we enumerate in Table 1. For eight of these tests, there is a fairly negligible difference (less than 10%). However, four of these tests have a fairly significant difference, with the top sites performing significantly better in these tests in comparison to the longtail sites.

First, we have strict-transport-security with 40% of top sites passing and only 14.99% of longtail sites passing. This test requires the HTTP Strict Transport Security (HSTS) response header to have been set by the site for a minimum of at least six months. One potential reason for why top sites perform better is that top sites have been quicker to adopt HTTPS, which is a logical prerequisite to using HSTS, since HSTS will cause the site to be loaded over HTTPS instead of HTTP on future visits. We saw evidence for quicker HTTPS adoption by top sites in the related work by Mirian et al. [3].

The remaining three tests are all related to "X-" headers, which used to be reserved for custom proprietary headers until 2012. The x-content-type-options test has 45% of top sites passing and only 23.43% of longtail sites passing. This test requires the X-Content-Type-Options header to be set to "nosniff," which reduces the XSS attack vector. The x-frame-options test has 64% of top sites passing and only 24.06% of longtail sites passing. This test requires the X-Frame-Options header to be configured correctly. This had the largest disparity in terms of passing rate, with nearly a 30% difference. Lastly, the x-xss-protection test has 32% of top sites passing and only 16.81% of longtail sites passing. This test requires the X-XSS-Protection header to be enabled (unless CSP covers it). One hypothesis to explore for why popular sites perform better on the "X-" header tests is that popular sites are more likely to use web hosting solutions that understand best practices associated with experimental headers.

Table 2: Performance of Top 10 Most Frequent Servers

Rank	Server	Avg Score	Frequency
1	cloudfare	13.67	2117
2	apache	8.24	1748
3	nginx	11.85	1615
4	no-server-header	14.23	1557
5	litespeed	12.32	365
6	microsoft-iis/10.0	8.81	193
7	openresty	8.35	176
8	microsoft-iis/8.5	8.60	118
9	apache/2	8.14	70
10	sucuri/cloudproxy	18.79	66
14*	squarespace	0.00	49

* Notable data point included for discussion.

5.3 Security Score by Server

In this section, we will look at how Mozilla Observatory score varies depending on the web server hosting the site. We found the web server using the self-reported "Server" HTTP response header field. Since this is reported by the site we are querying, this does not guarantee that the specified server was used, and we do not do any post-processing on the server field beyond lowercasing and trimming whitespace (i.e., we do not remove reported version numbers, so apache and apache/2 are different). We use the same 1% random sample from the Tranco list that we used in the previous "All Sites" analysis.

The most popular servers were cloudfare (2117), apache (1748), and nginx (1615). These three servers represent 54.80% of all of the servers in this random sample. Since the sample is random, we can assume that the majority of the servers in the Tranco Top 1M use one of these three servers. With 1557 occurrences, the fourth most commonly reported Server header was to omit the Server value from the HTTP response header (this is an overestimate, since this will include sites that missed our request).

Noting Table 2, apache had below average performance (more than 3 points below the full dataset's average score of 11.57 which we saw in the All Sites section), nginx was about average (11.85), and cloudfare was slightly better than average (13.67). Sites which did not report a server did the best of these four with 14.23. None of these servers averaged better than the top 10,000 sites (23.45). One more interesting result in this analysis was that squarespace was reported 49 times and averaged a score of 0.

When considering which servers performed the best, the results are listed in Table 3. To keep the results interesting, we limited the results to servers that were reported more than five times. Google Web Server (gws) was the "most secure" with an average score of 47.31 in 13 occurrences (47.31, 13). This is proprietary web server software used by Google to provide

Table 3: Top 10 Servers With Best Security Score

Rank	Server	Avg Score	Frequency*
1	gws	47.31	13
2	vercel	34.55	22
3	cloudfront	34.44	9
4	netlify	34.29	35
5	envoy	30.83	6
6	google frontend	29.55	11
7	nginx-rc	25.56	9
8	ats	25.38	13
9	apache/2.4.46 (ubuntu)	20.00	6
10	sucuri/cloudproxy	18.79	66

* Entries with less than 5 occurrences were excluded

web hosting services. Vercel (34.55, 22), cloudfront (34.44, 9), and netlify (34.29, 35) take the next three ranks and are also proprietary web hosting solutions. The top four servers all have average scores well above the average score for the top 10,000 sites (23.45). The 5th most secure server is envoy (30.83, 6), which is an open source web hosting solution.

6 Future Work

Mozilla Observatory's security score is one example of a security indicator, and it is what we used to answer the question of whether popularity indicates security. However, this is not the only indicator of security. Any reasonable security indicator could be used to repeat the same study and contribute to the discussion. For example, we had considered using the number of CVEs a website's server is vulnerable to as a discrete-valued security indicator (similar to Mozilla's score, but with lower values being better in this case). We found that it is straightforward to extract a website's server version from response headers. However, it was difficult to find which CVEs were associated with a particular server version, since this information was often written in a variety of formats using plain English within the CVE description.

Future work may also opt to repeat the same analysis with a larger percentage of the Tranco Top 1M. We chose 1% primarily from a feasibility standpoint, but it is not completely unreasonable to do a full scan of the Tranco list, though this should probably be done over a longer period of time (e.g. a week) to prevent significant stress on Mozilla Observatory. This would enable us to get better server-specific data in particular, where we saw a lot of servers with single to double digit frequencies, which did not feel very significant.

To "pass" a Mozilla Observatory test, it is typically sufficient to meet one of many conditions, but only of those conditions will reward bonus points and some conditions are arguably more secure than others. Future work may look at final test states as a form of deeper granularity (e.g. how many tests passed by finishing in one state compared to another). In addition, the scoring module for Mozilla Observatory does not seem very appropriate for research. The maximum amount of points you can currently obtain is 135, but a site with the worst possible configuration would receive a score of -160 if the minimum score was not floored to 0. As a result, there is actually quite a lot of variance in potential scores (-160 to 135) that is lost by setting the minimum score to 0 (enforcing 0 to 135 as the range). This is likely why the average Mozilla Observatory scores tended to be fairly low in our analysis. Future work should definitely look into revamping the scoring module to take into account all of the variance associated with sites that score below 0.

7 Ethics Consideration

Mozilla Observatory is a public tool meant to help websites identify vulnerabilities. One potential ethical concern could be that repeated querying of a site during the data collection phase could pose a burden to the sites being queried (or to Mozilla Observatory itself). To this end, we made sure to cache results from Mozilla Observatory and we introduced a long waiting phase into our data collection pipeline to ensure there was a very high probability that our scans were complete when we queried Mozilla Observatory (otherwise we would have to query repeatedly). On Mozilla Observatory's end, they cache scan results from the past 24 hours and take other measures to ensure they do not pose a significant burden on sites. The vast majority of the tests performed by Mozilla Observatory appear to simply check the response headers from the site being tested; there is no illegal aspect to it (such as running an XSS or SQLi attack).

8 Conclusion

Using the Mozilla Observatory score as a metric for security, there is a weak but significant indication towards website popularity being an indicator of better security. In our analysis, we saw that the average score for popular sites was nearly double the average score for all other sites. In addition, we saw a statistically significant but weak correlation coefficient in favor of popular sites being more secure. From an analysis of the web servers used, we saw that cloudfare, apache, and nginx composed the majority of all web servers, but the most secure web servers were proprietary web servers provided by web hosting solutions such as Google Web Server, Vercel, Cloudfront, and Netlify. We caution again that the Mozilla Observatory score loses a significant amount of variance due to the existing scoring methodology, and future work may get significantly more mileage by modifying the existing Mozilla Observatory score to accommodate the full range of potential scores (-160 to 135 as opposed to 0 to 135).

References

[1] Victor Le Pochat, Tom Van Goethem, Samaneh Tajalizadehkhoob, Maciej Korczynski, and Wouter Joosen. Tranco: A research-oriented top sites ranking hardened against manipulation. *Proceedings 2019 Network and Distributed System Security Symposium*, 2019.

[2] Mozilla. Website security test. Available at https://observatory.mozilla.org/ (visited on 02/01/2022).

[3] Ariana Mirian, Christopher Thompson, Stefan Savage, Geoffrey M. Voelker, and Adrienne Porter Felt. Https adoption in the longtail. Technical report, Google and UC San Diego, 2018.

[4] Miltiadis Siavvas, Marija Jankovic, Dionysios Kehagias, and Dimitrios Tzovaras. Is popularity an indicator of software security? In *2018 International Conference on Intelligent Systems (IS)*, pages 692–697, 2018.

[5] James Walden, Maureen Doyle, Grant Welch, and Michael Whelan. Security of open source web applications. pages 545–553, 10 2009.