Exploring Common Web Vulnerabilities through Static Analysis with Semgrep

Ahmet Eren Gündoğdu

Advisor: Dr. Ebu Yusuf Güven

Contents

Abstract

This study allows us to find the most common web vulnerabilities by performing static analysis. It uses WebScan, a Python-based tool to download source files and analyse them using Semgrep. The outputs obtained from the analyses are compiled to find most common vulnerabilities on web applications.

1. Introduction

The growth of web applications has made it difficult to prevent cyber threats. Static analysis tools are effective in detecting vulnerabilities in source code before they are exploited. In this study, we used WebScan, a python tool designed to fetch and analyse web applications source files. This project aims to analyse vulnerabilities in one hundred different websites and gain insights into the most common vulnerabilities in modern web development.

Businesses and individuals increasingly rely on web-based services over time and security of these applications has become paramount. Consequences of vulnerabilities for businesses are data breaches, financial loss, and reputational damage. Therefore, early detection and correction of vulnerabilities are critical in the software development lifecycle (SDLC) (14,15).

Previous studies have demonstrated the effectiveness of static analysis tools in uncovering common security vulnerabilities. For example, Livshits and Lam's research (10) utilized static analysis to detect vulnerabilities in Java applications, revealing numerous instances of SQL injection and cross-site-scripting (XSS) flaws. Similarly, Xie and Aiken (11) developed a static analysis framework that can identify resource leaks and inconsistent code.

However, relying on default configuration on static analysis tools may result in missed vulnerabilities. Johnson et al. (12) highlighted static analysis tools performance and effectiveness can be maximized when they configured properly. Additionally, automated tools with adequate user interface can be more appealing to the developers. Combining dynamic and static analysis can provide a more comprehensive security assessment, as dynamic analysis can uncover runtime vulnerabilities that static analysis might miss (13).

2. Methodology

2.1 Simple Random Sampling

In this research we selected 100 different websites amongst 500 most popular websites of 2024 and analyse them to find most common vulnerabilities. We employed Simple Random Sampling (SRS) method to select our subset of 100 websites. Simple Random Sampling is a type of probability sampling that gives each member of the selection set an equal chance of being selected.

First, we define the total population of websites. The population for this study includes 500 most popular websites, determined by "Moz"(6) which uses domain authority (7) to find most popular websites. Each website assigned to a number ranging from 1 to 500. Using a random number generator, 100 unique numbers were generated, and corresponding websites were selected.

We used Python's build-in function as a random number generator. The "random.sample()" function is in "random" module is designed to return a particular length list of items chosen from the sequence (i.e. list, tuple, string or set) (8,16).

2.2 WebScan Tool

WebScan automatically retrieves the website source files. After retrieving source files, there are stored in a structured directory as a JSON file, edits them for both static and dynamic analysis. It uses Semgrep for static analysis and OWASP ZAP, Nmap, SQLMap for dynamic analysis. LaTeX is used for creation of pdf report.

2.2.2 WebScan Software Architecture

In this section, our project's top-level software architecture will be shown and explained regarding their roles and interconnections.

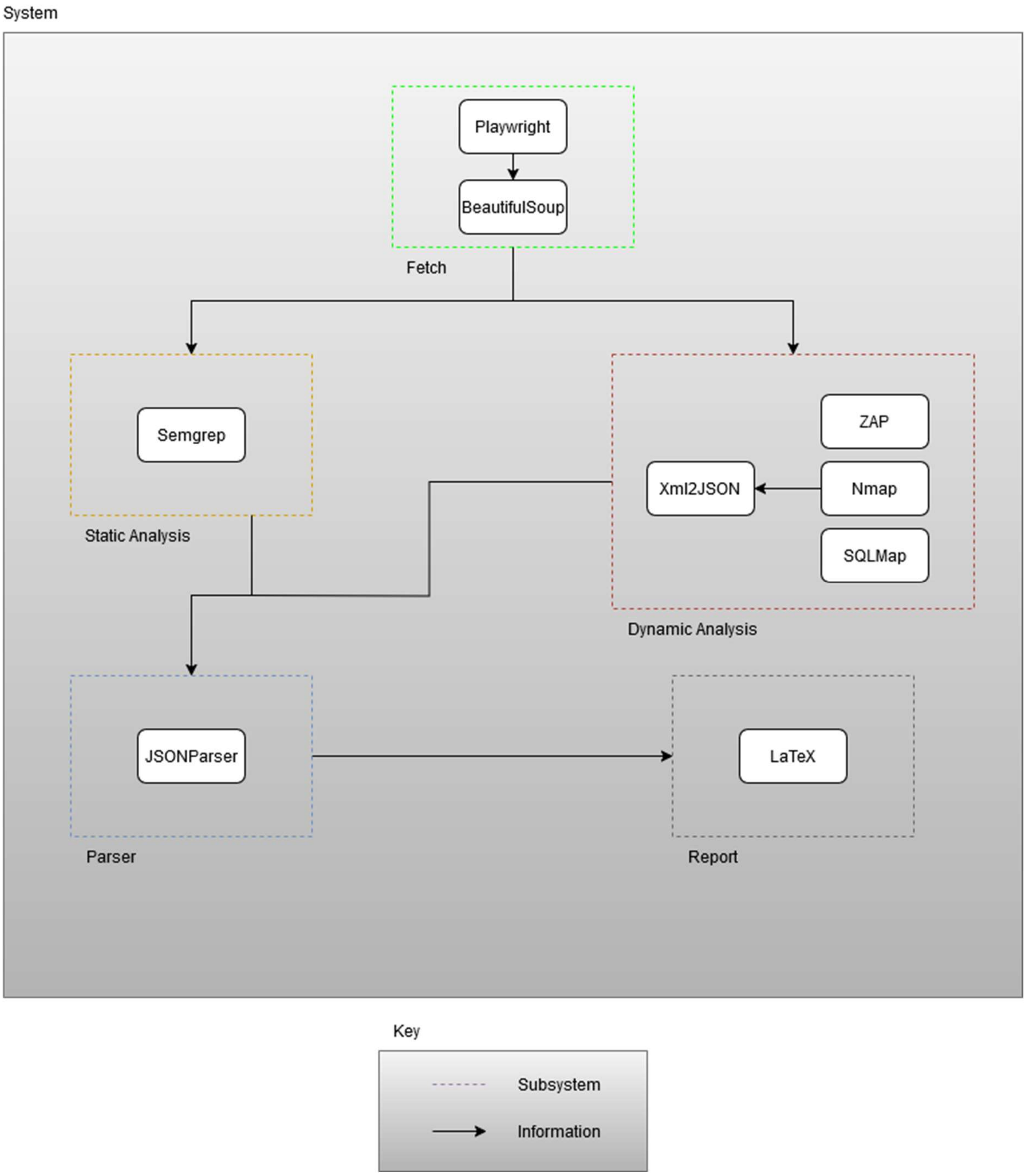


Figure 1 Software Architecture

2.2.3 Fetch

There are 2 steps to the first part of the WebScan tool, which is the fetch of the source files of the web application. The first of these steps is to open a chromium browser using Playwright and go to the target URL. In the second step, the source files of the target URL and its sub-URLs are fetched using BeautifulSoup. This process is done asynchronously, the advantage of doing it asynchronously is that the source files can be fetched more quickly and efficiently.

2.2.4 Static Analysis

Static Analysis is the automated analysis of source code without executing the application. The static analysis part of WebScan employing Semgrep. Semgrep is an open-source static analysis tool. It analyses the fetched files using Semgrep's default ruleset and outputs the results as a JSON file.

2.2.5 Dynamic Analysis

Dynamic analysis uses real-time data to assess a program and provide valuable insights on vulnerabilities that static analysis alone may miss. Dynamic analysis feature does not used in this project. Webscan performs dynamic analysis using Nmap, SQLMap, and OWASP ZAP.

2.2.6 Parser

Parsing is a technique used to transform a string containing a JSON document into a structured data object that we can operate on. If the outputs of the tools used for static and dynamic analysis are not in JSON format, they are converted to JSON and processed using parsing methods to make them useable for reporting.

2.2.7 Report

WebScan using LaTeX for reporting static and dynamic analysis findings on pdf format. LaTeX is a typesetting system includes features designed to produce technical and scientific documentation.

2.3 Workflow

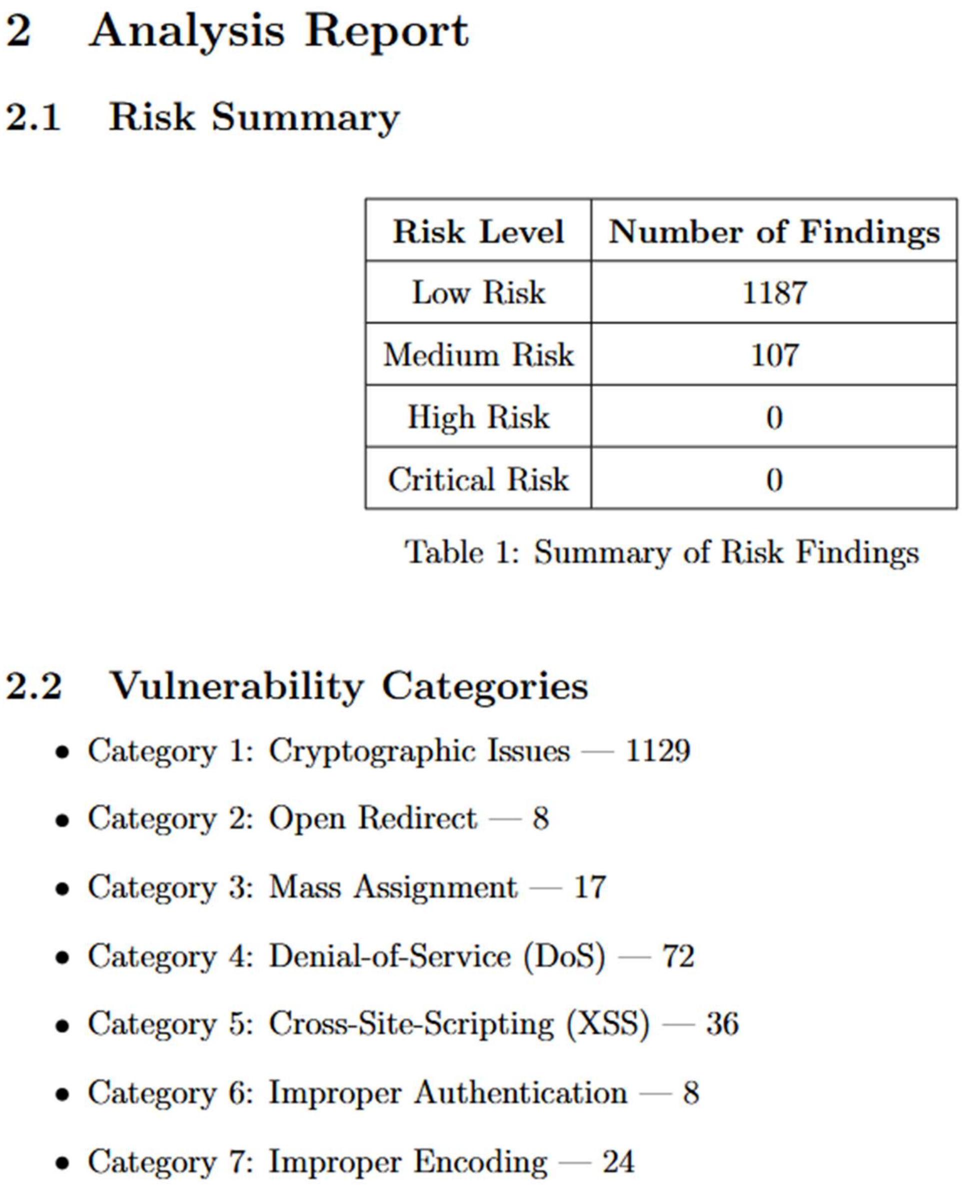
The workflow of this study begins with the selection of websites. A sample of 100 websites was chosen from the top 500 most popular websites of 2024 using the Simple Random Sampling (SRS) method. This approach ensured an unbiased representation of the population and provided a diverse dataset for analysis.

Next, the source files of the selected websites were fetched using the WebScan tool. This step involved automating browser interactions through Playwright to access the target websites and their subpages. The fetched files, including HTML, CSS, and JavaScript, were retrieved asynchronously using BeautifulSoup, enabling efficient and quick data collection.

Once the source files were collected, static analysis was performed using Semgrep. This open-source static analysis tool applied its default ruleset to identify potential vulnerabilities within the retrieved files. The output of this analysis was generated in JSON format, providing structured and detailed results.

Following the analysis, the results were parsed and processed into a unified format for reporting. Any non-JSON outputs were converted into JSON to ensure consistency. Using LaTeX, these findings were compiled into a professionally formatted PDF report, detailing the vulnerabilities and their associated risk levels.

Finally, the collected data was analysed to identify the most frequently occurring vulnerabilities and their classifications by risk type and severity. This comprehensive workflow ensured that the study effectively captured and analysed the vulnerabilities present in the selected websites.



2 Analysis Report

2.1 Risk Summary

Risk Level	Number of Findings
Low Risk	1187
Medium Risk	107
High Risk	0
Critical Risk	0

Table 1: Summary of Risk Findings

2.2 Vulnerability Categories

- Category 1: Cryptographic Issues — 1129
- Category 2: Open Redirect — 8
- Category 3: Mass Assignment — 17
- Category 4: Denial-of-Service (DoS) — 72
- Category 5: Cross-Site-Scripting (XSS) — 36
- Category 6: Improper Authentication — 8
- Category 7: Improper Encoding — 24

Figure 2 Analysis Report Created by Webscan for dropbox.com

Figure 2 shows a WebScan report created automatically for “dropbox.com”. All 100 different websites reports are created in the same manner and analysed Risk Summary section shows the identified risk levels, while the Vulnerability Categories header lists all instances of the detected vulnerability types.

3. Results and Analysis

This section of the study contains analysis of the results that obtained by static analysis.

The definition of risk in cybersecurity is the potential for exposure or loss resulting from a cyberattack or data breach on organizations or users. In cybersecurity, the risk levels of vulnerabilities are calculated by considering degree of vulnerability, and the likelihood of exploitation.

Semgrep relies on custom ruleset that can be configured by developers, but the default ruleset is using the OWASP Top Ten, focusing on the most critical and common web application security risks.

Risk levels and their occurrences is shown in table 1.

Table 1 Risk Level Table

Risk Level	**Number Of Findings**
Low Risk	104368
Medium Risk	15212
High Risk	31
Critical Risk	0

Results are shown that most common vulnerabilities are Cross-Site Scripting (XSS), Denial-of-Service (DoS), and Mishandled Sensitive Information. Although these vulnerabilities do not count as high or critical risk developers should take cautious against them.

Table 2 Vulnerability Categories

Vulnerability Type	**Number of Instances**
Cryptographic Issues	89171
Cross-Site Request Forgery (CSRF)	1010
Cross-Site-Scripting (XSS)	5151
Denial-of-Service (DoS)	8631
Improper Authentication	2766
Mishandled Sensitive Information	7750
Mass Assignment	1448
Improper Encoding	887
Improper Validation	1308
Code Injection	703
Open Redirect	345
Hard-coded Secrets	485

SQL Injection	1

Table 3 Domains with Most Vulnerabilities

Domain	Vulnerability Type 1	Vulnerability Type 2	Vulnerability Type 3
www.e-monsite.com	Cryptographic Issues — 23859	Mishandled Sensitive Information — 535	CSRF — 142
www.4shared.com	Cryptographic Issues — 19995	CSRF — 324	Denial-of-Service (DoS) — 74
www.spiegel.de	Cryptographic Issues — 6796	Improper Validation — 252	Mishandled Sensitive Information — 202
www.marca.com	Cryptographic Issues — 5998	Mishandled Sensitive Information — 616	Denial-of-Service (DoS) — 189
github.com	Cryptographic Issues — 4407	Cross-Site-Scripting (XSS) — 1405	Denial-of-Service (DoS) — 578
www.apple.com	Cryptographic Issues — 5114	Denial-of-Service (DoS) — 305	Cross-Site-Scripting (XSS) — 287
www.elmundo.es	Mishandled Sensitive Information — 3243	Cryptographic Issues — 818	Denial-of-Service (DoS) — 14
www.tradingview.com	Cryptographic Issues — 1926	Denial-of-Service (DoS) — 159	Mass Assignment — 79
www.temu.com	Cryptographic Issues — 1603	Hard Coded Secrets — 49	Mishandled Sensitive Information — 6
www.livejournal.com	Cryptographic Issues — 1229	Mishandled Sensitive Information — 128	CSRF — 75

Cryptographic issues emerged as the most frequently occurring vulnerabilities in our analysis. These issues are generally symptoms of a deeper underlying root cause rather than standalone problems. Determining the root cause is the key to solve these problems.(9)

XSS can be used by attackers to inject malicious scripts into web pages. These scripts can steal session cookies, redirect users to malicious websites, or perform actions on behalf of the victim. Input validation, output encoding, and implementing Content Security Policy (CSP) are common methods for defending against XSS. Additionally, using secure frameworks that automatically escape user input can significantly reduce the risk of XSS attacks (3).

DoS is a type of cyber-attack in which attackers aim to disrupt or shut down a targeted server, service or network by overwhelming it with a flood of requests that cause a crash. The target becomes slow, unresponsive, or utterly inaccessible to legitimate users. This attack can cripple websites, disrupt services, and cause financial and reputational loss (4). To defend against DoS attacks, implementing rate limitation, firewalls, and cloud-based DDoS protections can be used. Additionally, optimization of the system for high traffic loads and using load balancers can provide further protection.

Mishandled Sensitive Information vulnerabilities occur when sensitive data is mishandled or not protected correctly. Passwords, API keys or personal information can be given as examples of sensitive information. When these sensitive informations are not protected it can lead to unauthorized access and data breaches. Encryption for data both in transmit and at rest is the key for protecting sensitive information. Additionally, implementation of proper access controls, and ensuring that sensitive data is not stored or logged in database are valid methods against MSI (2,5).

By addressing these vulnerabilities with appropriate methods, organizations can reduce their risk of exploitation, even if these vulnerabilities are not considered the highest risk on their own.

4. Discussion

The obtained results signify how common are the basic but critical vulnerabilities in web applications. Even though the cryptographic issues are the most common vulnerabilities, the high amounts of XSS and DOS vulnerabilities are the indicators of lack of input validation and poor resource handling. The high occurrence of hardcoded secrets also shows the need for better secure coding practices that utilize more secure methods like key encryption. The use of outdated libraries increases the attack possibility for known vulnerabilities, thus should be avoided.

4.1 Limitations

Our study only works on publicly available website source files therefore we can’t analyse server-side vulnerabilities. WebScan is using default Semgrep configuration rules which can miss specific vulnerabilities.

5. Conclusion

Static analysis is an effective tool to detect vulnerabilities in an early stage. This study identified common vulnerabilities using the WebScan tool and provides recommendations for developers to mitigate web application vulnerabilities. WebScan uses Semgrep for static analysis. We identified most common vulnerabilities by analysing the source files of randomly selected 100 popular websites. Our findings indicates that most common vulnerabilities are Cross-site Scripting (XSS), cryptographic issues, and Denial-of-Service (DoS). Most common vulnerability is cryptographic issues. The results highlight the importance of using static analysis in early stages of software development lifecycle (SDLC) to prevent vulnerabilities before they are exploited. With 104,368 low-risk, 15,212 medium-risk and, 31 high-risk findings show the tools ability to detect many different instances of vulnerabilities. However, the study acknowledges the limitations of relying solely on Semgrep's default configuration, which may miss specific vulnerabilities.

Developers should improve in security practices in the fields of cryptographic implementations, input validation, and resource handling to prove greater protection against most common vulnerabilities.

6. References

1. The most popular websites: Top websites in 2024. Backlinko. Retrieved December 26, 2024, from https://backlinko.com/most-popular-websites
2. Wikina SB. What caused the breach? An examination of use of information technology and health data breaches. Perspect Health Inf Manag. 2014 Oct 1;11(Fall):1h. PMID: 25593574; PMCID: PMC4272442.
3. Hydara, I., Sultan, A. B. M., Zulzalil, H., & Admodisastro, N. (2015). Current state of research on cross-site scripting (XSS): A systematic literature review. Information and Software Technology, 58, 170–186. https://doi.org/10.1016/j.infsof.2014.07.010
4. Needham, R. M. (1993). Denial of service. In Proceedings of the 1st ACM Conference on Computer and Communications Security (CCS '93) (pp. 151–153). Association for Computing Machinery. https://doi.org/10.1145/168588.168607Mishandled Sensitive Information reference
5. R. Barona and E. A. M. Anita, "A survey on data breach challenges in cloud computing security: Issues and threats," 2017 International Conference on Circuit ,Power and Computing Technologies (ICCPCT), Kollam, India, 2017, pp. 1-8, doi: 10.1109/ICCPCT.2017.8074287.
6. Moz. Top 500 sites on the internet. Retrieved November 15, 2023, from https://moz.com/top500

7. Moz. Domain authority: What is domain authority? Retrieved from https://moz.com/learn/seo/domain-authority
8. GeeksforGeeks. Python random sample() function. Retrieved from https://www.geeksforgeeks.org/python-random-sample-function/
9. Livshits, V. B., & Lam, M. S. (2005). Finding security vulnerabilities in Java applications with static analysis. In Proceedings of the 14th USENIX Security Symposium (pp. 271-286).
10. Xie, Y., & Aiken, A. (2006). Static detection of security vulnerabilities in scripting languages. In Proceedings of the 15th USENIX Security Symposium (Vol. 15, pp. 179-192).
11. Johnson, B., Song, Y., Murphy-Hill, E., & Bowdidge, R. (2013). Why don't software developers use static analysis tools to find bugs?. In Proceedings of the 2013 International Conference on Software Engineering (pp. 672-681). IEEE Press.
12. Ernst, M. D. (2003, May). Static and dynamic analysis: Synergy and duality. In WODA 2003: ICSE Workshop on Dynamic Analysis (pp. 24-27).
13. Mosulet, P.-P. (2021). An Analysis of the Usage and Impact of Static Code Analysis Tools (Dissertation). Retrieved from https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-307412
14. AlBreiki, H. H., & Mahmoud, Q. H. (2014). Evaluation of static analysis tools for software security. In 2014 10th International Conference on Innovations in Information Technology (IIT) (pp. 93–98). IEEE.
15. Noor, S. , Tajik, O. , & Golzar, J. (2022). Simple Random Sampling. International Journal of Education & Language Studies, 1(2), 78-82. doi: 10.22034/ijels.2022.162982

7. Appendix

1. Semgrep's ruleset: https://github.com/semgrep/semgrep/blob/develop/semgrep.yml

Selectec Domains

1. google.com
2. youtube.com
3. reddit.com
4. bing.com
5. x.com
6. whatsapp.com
7. amazon.com
8. twitter.com
9. duckduckgo.com
10. tiktok.com

11. weather.com
12. microsoft.com
13. twitch.tv
14. office.com
15. vk.com
16. quora.com
17. discord.com
18. aliexpress.com
19. github.com
20. apple.com
21. spotify.com
22. roblox.com
23. cnn.com
24. nytimes.com
25. ebay.com
26. paypal.com
27. marca.com
28. espn.com
29. instructure.com
30. temu.com
31. zoom.us
32. rakuten.co.jp
33. dailymail.co.uk
34. dailymotion.com
35. theguardian.com
36. primevideo.com
37. disneyplus.com
38. as.com
39. usps.com
40. infobae.com
41. outlook.com
42. line.me
43. max.com
44. tradingview.com
45. blogger.com
46. myspace.com
47. weebly.com
48. dropbox.com
49. opera.com
50. ikea.com
51. mozilla.org

52. shopify.com
53. veja.abril.com.br
54. gravatar.com
55. brandbucket.com
56. netvibes.com
57. w3.org
58. jimdo.com
59. cpanel.net
60. nginx.org
61. mediafire
62. nature.com
63. hatena.ne.jp
64. 4shared.com
65. businessinsider.com
66. https://www.slideshare.net/
67. https://issuu.com/
68. webmd.com
69. https://elpais.com/
70. https://www.scribd.com/
71. https://www.forbes.com/
72. https://www.cnet.com/
73. https://thenai.org/
74. https://marketingplatform.google.com/about/
75. https://linktr.ee/
76. https://www.terra.com.br/
77. https://wordpress.com/
78. https://www.reg.ru/
79. https://www.dropcatch.com/
80. https://www.ovhcloud.com/
81. https://www.elmundo.es/
82. https://www.afternic.com/
83. https://www.huawei.com/
84. https://www.youronlinechoices.com/
85. https://youradchoices.com/
86. https://www.addtoany.com/
87. https://www.lemonde.fr/
88. https://www.dw.com/
89. https://www.spiegel.de/
90. https://imageshack.com/
91. https://theconversation.com/europe
92. https://www.leparisien.fr/

93. https://www.strikingly.com/
94. https://www.namecheap.com/
95. https://www.stanford.edu/
96. https://www.ign.com/
97. https://www.yelp.com/
98. https://www.zendesk.com/#
99. https://www.sagepub.com/
100. https://www.ziddu.com/
101. https://www.e-monsite.com/
102. https://www.livejournal.com/