Static and Dynamic Analysis

ISTEC-Cyber Security

November 8, 2024

Description

This report contains static and dynamic analysis of the target. It uses Semgrep for static analysis and OWASP ZAP, Nmap, and SQLMap for dynamic analysis.

Provided by

1 Static Analysis

Details about static analysis...

2 Analysis Report

2.1 Risk Summary

Risk Level	**Number of Findings**
Low Risk	76
Medium Risk	47
High Risk	0
Critical Risk	0

Table 1: Summary of Risk Findings

2.2 Vulnerability Categories

- Category 1: Denial-of-Service (DoS) — 46
- Category 2: Improper Encoding — 7
- Category 3: Mass Assignment — 6
- Category 4: Cross-Site-Scripting (XSS) — 7
- Category 5: Cryptographic Issues — 54
- Category 6: Improper Authentication — 1
- Category 7: Cross-Site Request Forgery (CSRF) — 1
- Category 8: Mishandled Sensitive Information — 1

2.3 Vulnerabilities by Page

Vulnerability 1	
Path: ScrapedFiles/faq/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 2	
Path: ScrapedFiles/faq/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 3	
Path: ScrapedFiles/faq/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 4	
Path: ScrapedFiles/faq/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 5	
Path: ScrapedFiles/faq/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 6	
Path: ScrapedFiles/faq/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 7	
Path: ScrapedFiles/faq/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 8	
Path: ScrapedFiles/faq/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 9	
Path: ScrapedFiles/faq/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a 'b' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 10	
Path: ScrapedFiles/faq/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like 'innerHTML', 'outerHTML' or 'document.write' is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 11	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 141
End	line: 58 col: 1070
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 12	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 1070
End	line: 58 col: 1149
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 13	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 123 col: 3
End	line: 123 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 14	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 128 col: 3
End	line: 128 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 15	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 129 col: 3
End	line: 129 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 16	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 197 col: 5
End	line: 198 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 17	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 198 col: 14
End	line: 198 col: 690
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 18	
Path: ScrapedFiles/faq/main/faq.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 215 col: 1
End	line: 215 col: 687
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 19	
Path: ScrapedFiles/index/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 20	
Path: ScrapedFiles/index/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 21	
Path: ScrapedFiles/index/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 22	
Path: ScrapedFiles/index/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 23	
Path: ScrapedFiles/index/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 24	
Path: ScrapedFiles/index/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 25	
Path: ScrapedFiles/index/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 26	
Path: ScrapedFiles/index/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 27	
Path: ScrapedFiles/index/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a ‘b‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 28	
Path: ScrapedFiles/index/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like ‘innerHTML‘, ‘outerHTML‘ or ‘document.write‘ is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 29	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 141
End	line: 58 col: 1070
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 30	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 1070
End	line: 58 col: 1149
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 31	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 115 col: 3
End	line: 115 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 32	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 120 col: 3
End	line: 120 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 33	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 121 col: 3
End	line: 121 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 34	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 189 col: 5
End	line: 190 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 35	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 190 col: 14
End	line: 190 col: 645
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 36	
Path: ScrapedFiles/index/main/index.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 207 col: 1
End	line: 207 col: 642
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 37	
Path: ScrapedFiles/lessons/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 38	
Path: ScrapedFiles/lessons/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 39	
Path: ScrapedFiles/lessons/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 40	
Path: ScrapedFiles/lessons/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 41	
Path: ScrapedFiles/lessons/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 42	
Path: ScrapedFiles/lessons/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 43	
Path: ScrapedFiles/lessons/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 44	
Path: ScrapedFiles/lessons/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 45	
Path: ScrapedFiles/lessons/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a 'b' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 46	
Path: ScrapedFiles/lessons/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like 'innerHTML', 'outerHTML' or 'document.write' is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 47	
Path: ScrapedFiles/lessons/main/gumroad.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 6 col: 5
End	line: 6 col: 125
Message	User controlled data in methods like 'innerHTML', 'outerHTML' or 'document.write' is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 48	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 16 col: 5
End	line: 16 col: 74
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 49	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 19 col: 137
End	line: 19 col: 1066
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 50	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 19 col: 1066
End	line: 19 col: 1145
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 51	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 19 col: 1223
End	line: 19 col: 1338
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 52	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 324 col: 9
End	line: 324 col: 116
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 53	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 324 col: 116
End	line: 324 col: 204
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 54	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 324 col: 204
End	line: 324 col: 873
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 55	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 324 col: 873
End	line: 324 col: 1552
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 56	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 715 col: 3
End	line: 715 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 57	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 720 col: 5
End	line: 720 col: 137
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 58	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 745 col: 3
End	line: 745 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 59	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 746 col: 3
End	line: 746 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 60	
Path: ScrapedFiles/lessons/main/lessons.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 814 col: 5
End	line: 815 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 61	
Path: ScrapedFiles/lessons/main/overlay-f8f9015a9aabefa09736.js	
Vulnerability Class	['Improper Authentication']
Start	line: 1 col: 5004
End	line: 1 col: 5436
Message	No validation of origin is done by the addEventListener API. It may be possible to exploit this flaw to perform Cross Origin attacks such as Cross-Site Scripting(XSS).

Vulnerability 62	
Path: ScrapedFiles/login/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 63	
Path: ScrapedFiles/login/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 64	
Path: ScrapedFiles/login/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 65	
Path: ScrapedFiles/login/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 66	
Path: ScrapedFiles/login/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 67	
Path: ScrapedFiles/login/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 68	
Path: ScrapedFiles/login/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 69	
Path: ScrapedFiles/login/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 70	
Path: ScrapedFiles/login/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a 'b' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 71	
Path: ScrapedFiles/login/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like 'innerHTML', 'outerHTML' or 'document.write' is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 72	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 141
End	line: 58 col: 1070
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 73	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 1070
End	line: 58 col: 1149
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 74	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cross-Site Request Forgery (CSRF)']
Start	line: 84 col: 25
End	line: 95 col: 32
Message	Manually-created forms in django templates should specify a csrf_token to prevent CSRF attacks.

Vulnerability 75	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 115 col: 3
End	line: 115 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 76	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 120 col: 3
End	line: 120 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 77	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 121 col: 3
End	line: 121 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 78	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 189 col: 5
End	line: 190 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 79	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 190 col: 14
End	line: 190 col: 679
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 80	
Path: ScrapedFiles/login/main/login.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 207 col: 1
End	line: 207 col: 664
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 81	
Path: ScrapedFiles/pages/forms/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 82	
Path: ScrapedFiles/pages/forms/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 83	
Path: ScrapedFiles/pages/forms/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 84	
Path: ScrapedFiles/pages/forms/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	'"https://www.google.%/ads/ga-audiences".replace' method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 85	
Path: ScrapedFiles/pages/forms/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 86	
Path: ScrapedFiles/pages/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 87	
Path: ScrapedFiles/pages/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 88	
Path: ScrapedFiles/pages/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 89	
Path: ScrapedFiles/pages/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 90	
Path: ScrapedFiles/pages/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 91	
Path: ScrapedFiles/pages/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 92	
Path: ScrapedFiles/pages/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 93	
Path: ScrapedFiles/pages/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 94	
Path: ScrapedFiles/pages/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a ‘b‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 95	
Path: ScrapedFiles/pages/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like ‘innerHTML‘, ‘outerHTML‘ or ‘document.write‘ is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 96	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 141
End	line: 58 col: 1070
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 97	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 58 col: 1070
End	line: 58 col: 1149
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 98	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 155 col: 3
End	line: 155 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 99	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 160 col: 3
End	line: 160 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 100	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 161 col: 3
End	line: 161 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 101	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 229 col: 5
End	line: 230 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 102	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 230 col: 14
End	line: 230 col: 696
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 103	
Path: ScrapedFiles/pages/main/pages.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 247 col: 1
End	line: 247 col: 681
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 104	
Path: ScrapedFiles/pages/simple/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 10 col: 486
End	line: 10 col: 529
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 105	
Path: ScrapedFiles/pages/simple/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 27 col: 150
End	line: 27 col: 189
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 106	
Path: ScrapedFiles/pages/simple/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 28 col: 304
End	line: 28 col: 368
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 107	
Path: ScrapedFiles/pages/simple/main/analytics.js	
Vulnerability Class	['Improper Encoding']
Start	line: 37 col: 130
End	line: 37 col: 184
Message	‘"https://www.google.%/ads/ga-audiences".replace‘ method will only replace the first occurrence when used with a string argument ("%"). If this method is used for escaping of dangerous data then there is a possibility for a bypass. Try to use sanitization library instead or use a Regex with a global flag.

Vulnerability 108	
Path: ScrapedFiles/pages/simple/main/analytics.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 48 col: 629
End	line: 48 col: 650
Message	RegExp() called with a 'a' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 109	
Path: ScrapedFiles/pages/simple/main/conversion.js	
Vulnerability Class	['Mass Assignment']
Start	line: 3 col: 152
End	line: 3 col: 158
Message	Possibility of prototype polluting function detected. By adding or modifying attributes of an object prototype, it is possible to create attributes that exist on every object, or replace critical attributes with malicious ones. This can be problematic if the software depends on existence or non-existence of certain attributes, or uses pre-defined attributes of object prototype (such as hasOwnProperty, toString or valueOf). Possible mitigations might be: freezing the object prototype, using an object without prototypes (via Object.create(null)), blocking modifications of attributes that resolve to object prototype, using Map instead of object.

Vulnerability 110	
Path: ScrapedFiles/pages/simple/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 34 col: 97
End	line: 34 col: 134
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 111	
Path: ScrapedFiles/pages/simple/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 39 col: 1267
End	line: 39 col: 1331
Message	RegExp() called with a ‘a‘ function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 112	
Path: ScrapedFiles/pages/simple/main/conversion.js	
Vulnerability Class	['Denial-of-Service (DoS)']
Start	line: 79 col: 343
End	line: 79 col: 372
Message	RegExp() called with a 'b' function argument, this might allow an attacker to cause a Regular Expression Denial-of-Service (ReDoS) within your application as RegExP blocks the main thread. For this reason, it is recommended to use hardcoded regexes instead. If your regex is run on user-controlled input, consider performing input validation or use a regex checking/sanitization library such as https://www.npmjs.com/package/recheck to verify that the regex does not appear vulnerable to ReDoS.

Vulnerability 113	
Path: ScrapedFiles/pages/simple/main/conversion.js	
Vulnerability Class	['Cross-Site-Scripting (XSS)']
Start	line: 117 col: 382
End	line: 117 col: 399
Message	User controlled data in methods like 'innerHTML', 'outerHTML' or 'document.write' is an anti-pattern that can lead to XSS vulnerabilities

Vulnerability 114	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 15 col: 1
End	line: 15 col: 87
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 115	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 61 col: 141
End	line: 61 col: 1070
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 116	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 61 col: 1070
End	line: 61 col: 1149
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 117	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Mishandled Sensitive Information']
Start	line: 114 col: 33
End	line: 114 col: 161
Message	This link points to a plaintext HTTP URL. Prefer an encrypted HTTPS URL if possible.

Vulnerability 118	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3892 col: 3
End	line: 3892 col: 92
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 119	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3897 col: 3
End	line: 3897 col: 99
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 120	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3898 col: 3
End	line: 3898 col: 123
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 121	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3966 col: 5
End	line: 3967 col: 14
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 122	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3967 col: 14
End	line: 3967 col: 725
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

Vulnerability 123	
Path: ScrapedFiles/pages/simple/main/simple.html	
Vulnerability Class	['Cryptographic Issues']
Start	line: 3984 col: 1
End	line: 3984 col: 722
Message	This tag is missing an 'integrity' subresource integrity attribute. The 'integrity' attribute allows for the browser to verify that externally hosted files (for example from a CDN) are delivered without unexpected manipulation. Without this attribute, if an attacker can modify the externally hosted resource, this could lead to XSS and other types of attacks. To prevent this, include the base64-encoded cryptographic hash of the resource (file) you're telling the browser to fetch in the 'integrity' attribute for all externally hosted files.

3 Dynamic Analysis

Details about dynamic analysis...

4 Analysis Report

4.1 Nmap Scan Results

SQLMap Injection Points

Injection Point 1:

- **Parameter:** artist (GET)

    Type: boolean-based blind

    Title: AND boolean-based blind - WHERE or HAVING clause

    Type: time-based blind

    Title: MySQL ¿= 5.0.12 AND time-based blind (query SLEEP)

    Type: UNION query

    Title: MySQL UNION query (NULL) - 3 columns

4.2 Risk Summary

Risk Level	**Number of Findings**
Low Risk	zaplc
Medium Risk	zapmc
High Risk	zaphc
Critical Risk	zapcc

Table 2: Summary of Risk Findings

4.3 Vulnerability Categories

- ZapCategories:

4.4 Vulnerabilities by Page