

Project Report

TITLE:

**Analyzing the BYKEA Data Exposure: A
Case Study on Cloud Misconfiguration,
Data Governance, and Remediation
Strategies**

Submitted to Sir Ali Naseer

BY:

Hasan Raza

1.Introduction & Background (BYKEA Breach)

BYKEA, a leading Karachi-based ride-hailing and delivery platform, suffered a major data exposure incident in 2020 when its production database was found publicly accessible on an Elasticsearch server without authentication. Security researchers discovered over **200 GB of data**—containing **more than 400 million records**—including sensitive customer and driver information such as full names, phone numbers, email addresses, trip details, physical addresses, CNIC numbers, driver license details, and even internal employee login credentials stored in plaintext.

The exposed Elasticsearch instance was indexed on the open internet, meaning anyone using tools like Shodan could access it. This was not due to a sophisticated hack, but a **cloud misconfiguration** and lack of proper access control. Although BYKEA secured the database within 24 hours after notification, the data had already remained exposed for weeks, creating significant risk of misuse, identity theft, fraud, and privacy violations.

This case highlights critical issues in Information Assurance, including the dangers of insecure cloud deployments, improper data governance, storing sensitive data without encryption, and the absence of monitoring systems to detect publicly exposed services. The incident provides a strong foundation for analyzing threat models, vulnerabilities, and mitigation strategies in modern cloud-based platforms.

2.Threat Model of BYKEA Data Exposure Case Study

This section details the security risks associated with the BYKEA data exposure incident by applying structured threat modeling techniques, specifically the **STRIDE** methodology.

2.1 Scope and Assumptions

The threat model focuses on the core system component involved in the data exposure, as discovered by security researchers

Component	Description
Target System	BYKEA's production telemetry and records platform , primarily an Elasticsearch cluster or equivalent search/index service.
Deployment Context	An Internet-accessible Elasticsearch server indexing production data and API logs. Crucially, it lacked authentication, making it publicly

	accessible.
Attack Surface	The primary surface was the exposed Elasticsearch endpoint accessible from the Internet , discoverable via search tools like Shodan.
Assumption	The exposure resulted from misconfiguration (publicly readable instance) and default/insecure access controls, not an advanced zero-day exploit.

2.2 Key Assets Exposed

The value of the threat model is determined by the sensitivity of the assets at risk. The exposure involved roughly 200 GB of data, containing over 400 million records of highly sensitive information.

- **Customer PII:** Full names, email addresses, phone numbers, and detailed trip invoices (pick-up/drop-off locations, times, distances, fares).
- **Driver PII:** Full names, phone numbers, physical addresses, National ID (CNIC) **numbers**, driver license details (issuing city and expiry), and body temperature logs.
- **Internal Credentials:** Employee login credentials stored in plain text.
- **Metadata:** API logs and production host identifiers.

2.3 Representative Attack Path

The incident follows a predictable path of opportunistic data theft enabled by security misconfiguration. This path demonstrates how the **Untrusted Zone** successfully breached the **Semi-Trusted Zone** perimeter.

Step	Description	STRIDE Threat
1. Discovery	An external attacker or opportunistic scanner (e.g., using Shodan) discovers the exposed Elasticsearch instance on the public Internet.	Information Disclosure

2. Access	The server allows unauthenticated read and list access to indices containing production data.	Information Disclosure
3. Exfiltration	The attacker queries large datasets or downloads index snapshots, extracting customer and driver PII and internal plaintext credentials.	Information Disclosure
4. Persistence & Impact	The extracted data is exfiltrated and may be sold, leaked, or used for targeted fraud and identity theft. The exposure persists for weeks until external notification.	Repudiation

2.4 STRIDE Analysis

The STRIDE model (Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, Elevation of Privilege) was used to systematically identify threats and suggest mitigations for the BYKEA service.

Threat Category	Specific Threat (BYKEA Context)	Recommended Mitigation
Spoofing	An attacker impersonates API consumers or actors to masquerade as legitimate users relying on publicly available metadata.	Enforce authenticated access to all data services and do not accept unauthenticated connections from the Internet.
Tampering	If ACLs were misconfigured to allow write access, an attacker could alter or inject malicious records into the index.	Enforce strict write vs read ACLs and restrict write operations to known ingestion endpoints.

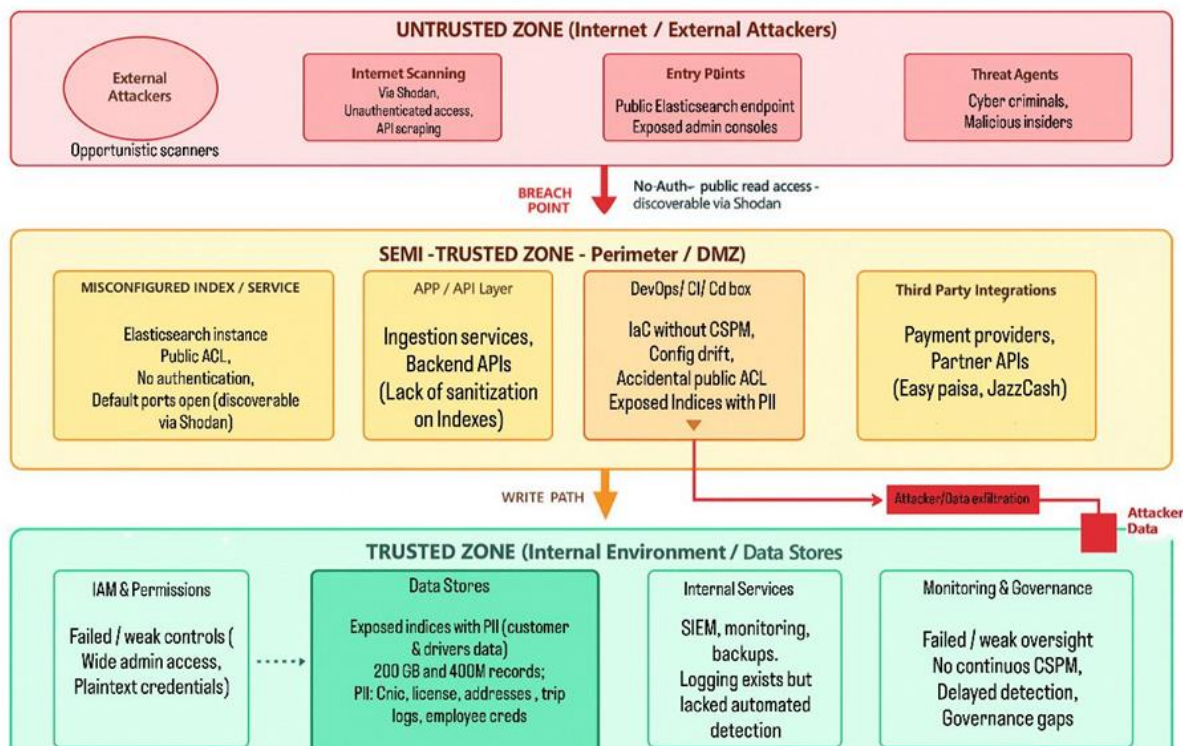
Repudiation	Incomplete audit trails on the exposed server make it difficult to prove or disprove data access or modifications.	Centralize immutable audit logs (append-only) and enable logging of all index access.
Information Disclosure	Unauthenticated access allowed the exfiltration of sensitive PII (customer/driver data) and employee credentials stored in plaintext.	Never store credentials in plaintext, enforce encryption at rest/in transit, and apply field-level redaction for PII.
Denial of Service	The publicly accessible search endpoint could be abused to overload resources through excessive queries or scraping.	Apply rate limiting, network ACLs, and capacity protections; run public queries through API gateways with quotas.
Elevation of Privilege	Overly broad admin credentials used by developers/CI accounts could enable attackers who gain limited access (e.g., via the index) to escalate privileges.	Enforce least privilege on administrative accounts and require MFA for console access.

2.5 Trust Boundaries and Visual Representation

The architecture is best understood by visualizing the failure of trust boundaries, specifically the breach point between the **Untrusted Zone** (Internet) and the **Semi-Trusted Zone** (DMZ/Perimeter).

The following diagram illustrates the flow of attack, data exfiltration, and the various misconfigured components (Misconfigured Index/Service, IAM/Permissions, etc.) that contributed to the breach.

The key takeaway is the failure of the boundary between the Internet and the exposed Elasticsearch endpoint, which was directly accessible without authentication. This misconfiguration created the "Breach Point."



3.Web Application VAPT via OWASP ZAP

This section details the Vulnerability Assessment and Penetration Testing (VAPT) performed on the public-facing web presence of the case study target, <https://bykea.com>, using **OWASP ZAP**. The test aimed to identify surface-level security misconfigurations and application weaknesses that could serve as initial entry vectors for a generalized attack, thereby supporting the original threat model.

3.1 Methodology and Scope

Parameter	Value	Justification / Context
Target URL	https://bykea.com (Main marketing site and public pages)	Testing was restricted to the public-facing web layer, and this test focuses on surface-level exposure.

Testing Type	Dynamic Application Security Testing (DAST)	Simulates an external, unauthenticated attacker interacting with the live site.
Scan Status	Partially Completed (10% Active Scan Progress)	The scan was stopped prematurely to adhere to responsible disclosure and minimize impact on the production server. The passive scan was fully utilized.
Key Risk Focus	Misconfiguration (OWASP A05:2021) and Broken Access Control (OWASP A01:2021).	Direct linkage to the root causes identified in the BYKEA case study proposal.

3.2 Report and Summary of Findings and Risk Analysis

The screenshot displays the ZAP (Zed Attack Proxy) interface during a manual exploration session. The top menu bar includes File, Edit, View, Analyse, Report, Tools, Import, Export, Online, and Help. The toolbar contains various icons for navigation and analysis. On the left, a 'Sites' pane lists several websites, including https://www.google.com.pk, https://www.google-analytics.com, https://connect.facebook.net, https://www.googletagmanager.com, https://ajax.googleapis.com, https://code.jquery.com, https://www.bykea.com, https://fonts.gstatic.com, https://fonts.googleapis.com, https://unpkg.com, https://challenges.cloudflare.com, and https://static.cloudflareinsights.com. The central pane is titled 'Manual Explore' and contains instructions for launching a browser through ZAP. It shows the URL to explore as https://bykea.com/.app and provides options to enable the HUD and launch the browser. A warning message states: 'You appear to be running ZAP in a container, so launching browsers has been disabled as this is unlikely to work.' Below this, it mentions the need for a browser that can proxy through ZAP and the ZAP root CA certificate. The bottom section features a progress bar for a new scan of https://bykea.com, showing 10% progress. Below the progress bar is a table of sent messages.

ID	Req. Timestamp	Resp. Timestamp	Method	URL	Code	Reason	RTT	Size Resp. Header	Size Resp. Body
29,616	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/challenge-platform/h/b/flo...	400	Bad Request	141 ms	1,296 bytes	14 bytes
29,617	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/rum?d+allow_url_include...	415	Unsupported Medi...	136 ms	252 bytes	579 bytes
29,618	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/rum?d+allow_url_include...	415	Unsupported Medi...	139 ms	252 bytes	579 bytes
29,619	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/challenge-platform/h/b/js...	200	OK	140 ms	1,828 bytes	0 bytes
29,620	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/?d+allow_url_include%3d1+d+au...	403	Forbidden	154 ms	2,293 bytes	7,074 bytes
29,621	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/challenge-platform/h/b/flo...	400	Bad Request	161 ms	1,290 bytes	14 bytes
29,622	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/rum?d+allow_url_include...	415	Unsupported Medi...	131 ms	252 bytes	579 bytes
29,623	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/challenge-platform/h/b/js...	200	OK	423 ms	1,821 bytes	0 bytes
29,624	11/22/25, 11:12:05 PM	11/22/25, 11:12:05 PM	POST	https://bykea.com/cdn-cgi/rum?d+allow_url_include...	415	Unsupported Medi...	140 ms	252 bytes	579 bytes



About this report

Report parameters

Contexts

No contexts were selected, so all contexts were included by default.

Sites

The following sites were included:

- <https://bykea.com>

(If no sites were selected, all sites were included by default.)

An included site must also be within one of the included contexts for its data to be included in the report.

Risk levels

Included: High, Medium, Low, Informational

Excluded: None

Confidence levels

Included: User Confirmed, High, Medium, Low

Excluded: User Confirmed, High, Medium, Low, False Positive

Summaries

Alert counts by risk and confidence

This table shows the number of alerts for each level of risk and confidence included in the report.

(The percentages in brackets represent the count as a percentage of the total number of alerts included in the report, rounded to one decimal place.)

		Confidence				
		User Confirmed	High	Medium	Low	Total
Risk	High	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
	Medium	0 (0.0%)	1 (8.3%)	1 (8.3%)	1 (8.3%)	3 (25.0%)
	Low	0 (0.0%)	1 (8.3%)	2 (16.7%)	1 (8.3%)	4 (33.3%)
	Informational	0 (0.0%)	0 (0.0%)	3 (25.0%)	2 (16.7%)	5 (41.7%)
	Total	0 (0.0%)	2 (16.7%)	6 (50.0%)	4 (33.3%)	12 (100%)

Alert counts by site and risk

This table shows, for each site for which one or more alerts were raised, the number of alerts raised at each risk level.

Alerts with a confidence level of "False Positive" have been excluded from these counts.

(The numbers in brackets are the number of alerts raised for the site at or above that risk level.)

		Risk			
Site		High (= High)	Medium (>= Medium)	Low (>= Low)	Informational (= Informational)
	https://bykea.com	0 (0)	2 (2)	3 (5)	3 (8)

Alert counts by alert type

This table shows the number of alerts of each alert type, together with the alert type's risk level.

(The percentages in brackets represent each count as a percentage, rounded to one decimal place, of the total number of alerts included in this report.)

Alert type	Risk	Count
Absence of Anti-CSRF Tokens	Medium	2 (16.7%)
Content Security Policy (CSP) Header Not Set	Medium	22 (183.3%)
Cross-Domain Misconfiguration	Medium	1 (8.3%)
Cookie with SameSite Attribute None	Low	5 (41.7%)
Cross-Domain JavaScript Source File Inclusion	Low	43 (358.3%)
Strict-Transport-Security Header Not Set	Low	13 (108.3%)
Timestamp Disclosure - Unix	Low	145 (1,208.3%)
Information Disclosure - Suspicious Comments	Informational	30 (250.0%)
Information Disclosure - Suspicious Comments	Informational	30 (250.0%)
Modern Web Application	Informational	22 (183.3%)
Re-examine Cache-control Directives	Informational	18 (150.0%)
Retrieved from Cache	Informational	44 (366.7%)
Session Management Response Identified	Informational	8 (66.7%)
Total		12

Alerts

Risk=Medium, Confidence=High (1)

<https://bykea.com> (1)

Content Security Policy (CSP) Header Not Set (1)

► GET <https://bykea.com/>

Risk=Medium, Confidence=Medium (1)

Risk=Medium, Confidence=Low (1)

<https://bykea.com> (1)

Absence of Anti-CSRF Tokens (1)

► GET <https://bykea.com/contact/>

Risk=Low, Confidence=High (1)

<https://bykea.com> (1)

Strict-Transport-Security Header Not Set (1)

► GET <https://bykea.com/>

Risk=Low, Confidence=Medium (2)

<https://bykea.com> (1)

Cross-Domain JavaScript Source File Inclusion (1)

► POST <https://bykea.com/>

Risk=Low, Confidence=Low (1)

<https://bykea.com> (1)

Timestamp Disclosure - Unix (1)

► GET <https://bykea.com/>

Risk=Informational, Confidence=Medium (3)

<https://bykea.com> (2)

[Modern Web Application](#) (1)

- ▶ GET <https://bykea.com/>

[Retrieved from Cache](#) (1)

- ▶ GET <https://bykea.com/wp-includes/css/dist/block-library/style.min.css?ver=6.1.7>

Risk=Informational, Confidence=Low (2)

<https://bykea.com> (1)

[Re-examine Cache-control Directives](#) (1)

- ▶ GET <https://bykea.com/about-us/>

The manual exploration yielded 12 total alerts, with 3 categorized as Medium Risk and 4 categorized as Low Risk. No High-Risk alerts were found, suggesting the site is not trivially vulnerable to common exploits like SQL Injection or XSS.

The primary risk vector identified is Security Misconfiguration, reinforcing a core finding of the BYKEA case.

Critical Findings (Medium Risk):

Finding	Severity (ZAP)	STRIDE Link	Impact & Relevance to BYKEA Case

Absence of Anti-CSRF Tokens	Medium Risk (Low Confidence)	Tampering	Found on a contact form (<code>/contact/</code>). This vulnerability allows an attacker to force a logged-in user (e.g., an internal employee or partner logged into a web portal) to submit unintended requests, potentially causing data modification or exposure. This is a classic Access Control failure.
Content Security Policy (CSP) Header Not Set	Medium Risk (High Confidence)	Information Disclosure	CSP is a defense-in-depth header that mitigates Cross-Site Scripting (XSS) and data injection. Its absence indicates a Security Misconfiguration that leaves users vulnerable to client-side attacks, which could be used to harvest session cookies or redirect users for phishing.
Cross-Domain Misconfiguration	Medium Risk (Low Confidence)	Information Disclosure	This suggests improper configuration of security headers (like CORS), which can unintentionally allow other domains to access resources from <code>bykea.com</code> , potentially leaking data or sensitive headers.

Secondary Findings (Low Risk - Focus on Misconfiguration)

Finding	Risk (ZAP)	OWASP/CWE Category	Relevance to BYKEA Case Study
Strict-Transport-Security Header Not Set	Low Risk	Security Misconfiguration (CWE-319)	The HSTS header forces browsers to use HTTPS, protecting against Man-in-the-Middle attacks that downgrade connections to insecure HTTP. Its absence is a foundational security lapse.
Re-examine Cache-control Directives	Informational /Low Risk (CWE-525)	Security Misconfiguration	Incorrect caching rules could allow unauthorized caching of sensitive information by local or proxy servers, creating an indirect Information Disclosure risk.

3.3 Linkage to Case Study and Mitigation

The VAPT confirms that while the application layer may be robust against simple injection attacks, the organization's official site exhibits critical Security Misconfiguration issues, a factor that was central to the 2020 BYKEA ElasticSearch exposure.

- **CSP/HSTS Absence:** The lack of these foundational security headers demonstrates that Security Guardrails and Automated Configuration Checks (as discussed in the proposal) were clearly missing or ineffective across the development and deployment pipelines.

- **CSRF Vulnerability:** The uncovered CSRF vulnerability, even on the public site, highlights a lack of continuous asset visibility and rigorous security review in development workflows, which correlates directly with the initial incident's root cause.

Mitigation Strategy (Proposed in Original Proposal)	VAPT Finding It Addresses
Embed automated configuration and compliance checks into developer/operations workflows	Directly addresses missing CSP and HSTS headers and poor cache control.
Implement strict default-deny access controls	The Absence of Anti-CSRF Tokens demonstrates an implicit trust in input that must be replaced by strict validation and tokenization to prevent Tampering.

3.4 Conclusion

The VAPT via ZAP successfully identified several Medium and Low-Risk security misconfigurations on the public web application. While the findings themselves do not point to the Elasticsearch data store, they provide concrete, actionable evidence of the governance and configuration control gaps that enabled the much larger data exposure in 2020. The remediation must focus on implementing modern security headers, cross-site request forgery prevention, and integrating automated DAST tools into the continuous integration (CI) workflow.

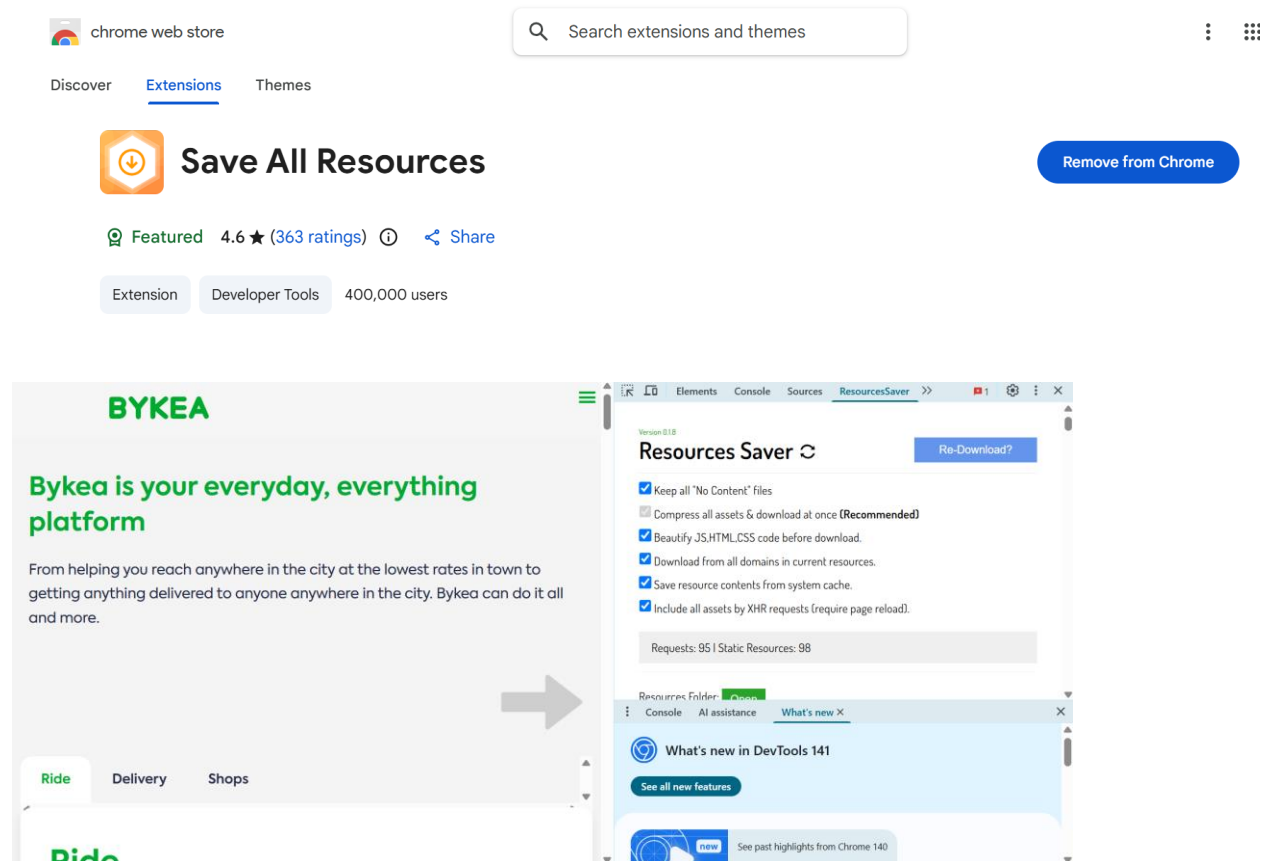
4. Code Acquisition for Static Analysis

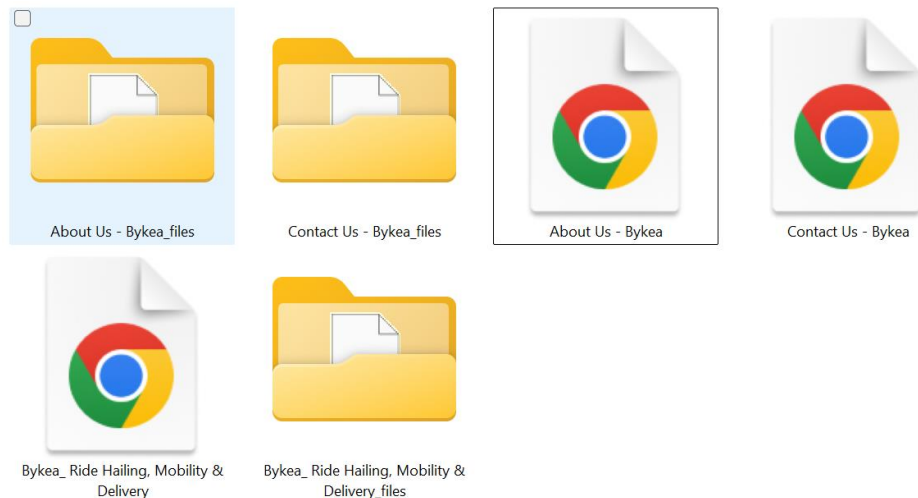
This phase of the project focuses on acquiring the necessary source code for **Static Application Security Testing (SAST)**. Given the architectural context of the BYKEA incident, the public web domain was analyzed to extract publicly exposed client-side code, which is highly relevant for hunting insecure components and hardcoded secrets.

4.1 Code Acquisition Methodology

The code acquisition was performed using a precise, non-aggressive scraping method, specifically targeting the application paths where the previous Dynamic Application Security Test (DAST) via ZAP raised security concerns.

- **Tool Used:** Standard, browser-based web scraping extension(Save All Resources) (*Right-Click On Page > Inspect > Resource Saver > Save All Button > complete*) was utilized. This safely collected the HTML structure and all supporting assets (CSS, images, and JavaScript files) without stressing the production environment.
- **Targeted Pages:** The acquisition was limited to the three pages identified in the VAPT with the highest-priority (Medium and Low Risk) alerts:
 - <https://bykea.com/> (Home Page)
 - <https://bykea.com/contact/>
 - <https://bykea.com/about-us/>
- **Collected Assets:** The scraping resulted in several HTML files and numerous JavaScript (.js) files bundled in their respective asset folders. These JavaScript files represent the executable client-side logic and form the target codebase for the subsequent SAST.





4.2 Strategic Rationale for SAST Target

While the core vulnerability of the original BYKEA breach was a **server-side misconfiguration** (exposed Elasticsearch), the collected client-side code is critical for demonstrating how foundational security controls could have prevented secondary exposures:

SAST Focus Area	Relevance to BYKEA Case Study	Target Files
Secrets Exposure	The original breach involved plaintext employee credentials. Client-side code is often mistakenly used to store hardcoded API keys or tokens.	Scraped JavaScript files
Vulnerable Dependencies (SCA)	Modern breaches exploit vulnerable third-party libraries. Scanning the JavaScript dependencies checks the integrity of the application's external components.	Scraped JavaScript files

Input Sanitization (XSS/DOM)	The absence of a Content Security Policy (CSP) headers indicated a high risk of Cross-Site Scripting (XSS). SAST can check the client-side code for vulnerable functions like eval() or innerHTML that enable DOM-based XSS.	Scraped JavaScript files and HTML
------------------------------	--	-----------------------------------

The next phase of the project will utilize this collected code base to run Snyk.io, demonstrating the application of static analysis to identify and remediate these critical code-level security issues before deployment.

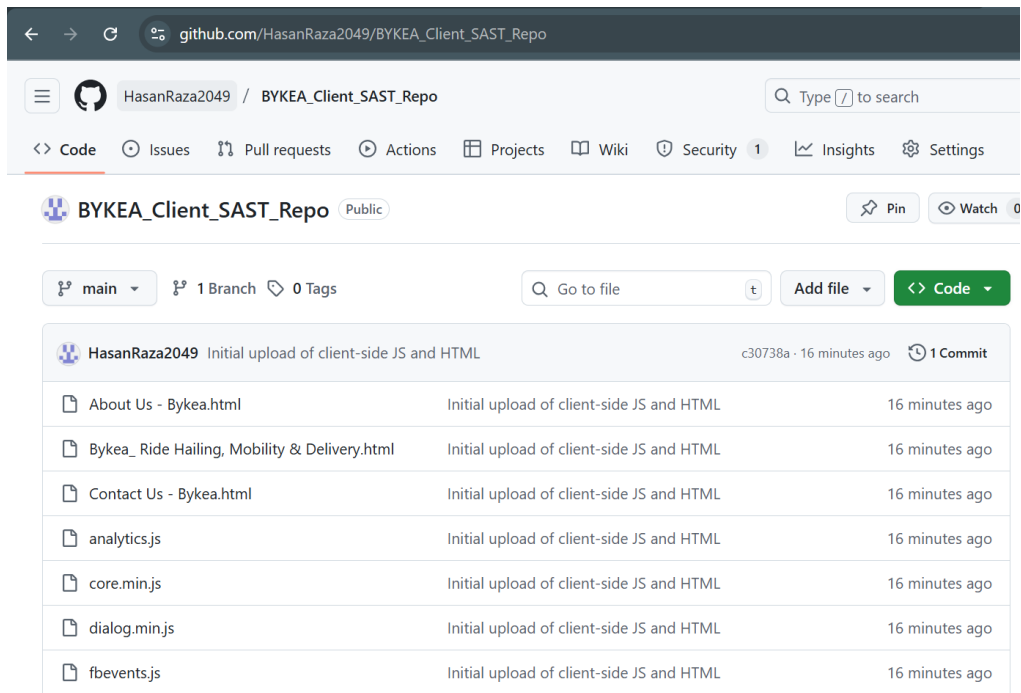
5. GitHub Repository Creation and Snyk Integration

This phase establishes the foundational environment for continuous security testing, directly addressing the governance and oversight failures identified in the original threat model.

The scraped client-side source code (HTML, JavaScript) was committed to a new repository on GitHub, github.com/HasanRaza2049/BYKEA_Client_SAST_Repo.

5.1 Repository Setup and Assets

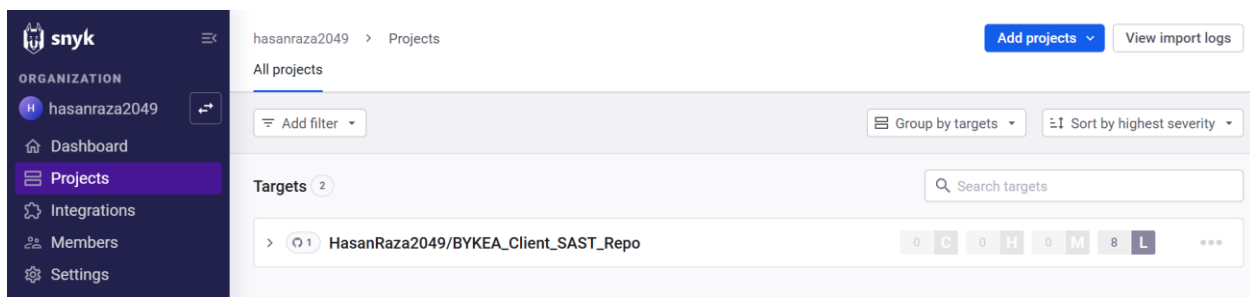
- **Repository Type:** The project code was consolidated into a new, single GitHub repository.
- **Assets Included:** Only executable and structural files were included: **HTML files** (entry points) and **JavaScript (.js) files** (client-side logic). Non-relevant files like CSS and image assets were excluded to maintain focus on security analysis.



5.2 Snyc Integration and Automated SAST

The connection between GitHub and Snyc was successfully established to automatically scan the project.

- **Action:** Snyc was authorized to access the repository and immediately initiated a scan (SAST and SCA) of the committed codebase.
- **Significance:** This integration demonstrates the implementation of **automated security checks into the developer/operations workflows**, serving as a core remediation strategy to prevent security issues like exposed secrets or vulnerable dependencies from reaching production, a critical failure point in the original BYKEA case.



6. Analysis of Snyk Findings and Security Integrity

This phase analyzed the client-side JavaScript and HTML for vulnerabilities using Snyk's SAST engine, confirming both the existing security posture of the public site and the utility of automated SAST in identifying remediation needs.

6.1 Scan Context and Tool Limitations

This comprehensive Snyk scan reported such less vulnerabilities due to the nature of the codebase:


- **Minified Code:** The vast majority of the uploaded JavaScript files were **minified, bundled, or obfuscated**. Static analysis tools like Snyk struggle to accurately trace data flow and identify complex vulnerabilities in such code.
- **Targeted Analysis:** To extract meaningful findings, the analysis was refined by focusing Snyk specifically on the **Google Tag Manager (GTM)** integration files, which often contain non-minified functional code that handles data exchange.

6.2 Results For The Scan:

The screenshot displays the Snyk Code Analysis interface. On the left is a dark sidebar with the Snyk logo and navigation links: Organization (hasanraza2049), Dashboard, Projects (selected), Integrations, Members, and Settings. The main content area shows the project details for 'HasanRaza2049/BYKEA_Client_SAST_Repo' (main branch). It includes tabs for Overview, History, and Settings. The Overview tab is active, showing a 'Code Analysis' section with a snapshot for commit b6f1cc6 taken by snyk.io. Below this, there are sections for 'Imported By' (hasanraza2049@gmail.com), 'Environment' (Add a value), and 'Lifecycle' (Add a value). On the right, there are sections for 'Project Owner' (Add a project owner), 'Business Criticality' (Add a value), and 'Analysis Summary' (57 analyzed files (93%), with a 'Repo breakdown' link). A modal window titled 'ANALYZED FILES' is open, showing a table with columns for file type and count: .html (3) and .js (54). Below this, it shows 'UNANALYZED FILES' with a count of 4.

File Type	Count
.html	3
.js	54

File Type	Count
Unanalyzed Files	4


 Code Analysis


OverviewHistorySettings

☐ Sensitive Cookie in HTTP... 4

L

Insufficient postMessage Validation




SNYK CODE | CWE-20 

SCORE
450

```
13320 |         var p = function() {
13321 |             e.contentDocument.body.appendChild(f);
13322 |             f.addEventListener("load", function() {
13323 |                 d.la = f.contentWindow;
13324 |                 e.contentWindow.addEventListener("message", function(q) {
```

The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

 gtmaboutus.js 

1 step in 1 file

[Learn about this type of vulnerability and how to fix it](#)

 Code Analysis

OverviewHistorySettings

L

Sensitive Cookie in HTTPS Session Without 'Secure' Attribute




SNYK CODE | CWE-614 

SCORE
450

```
9880 |     function Er(a, b, c, d) {
9881 |         var e = Ar(),
9882 |             f = window;
9883 |         wr(f) && (f.document.cookie = a);
```

Cookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

 gtmaboutus.js 

1 step in 1 file

 Code Analysis

OverviewHistorySettings

L

Insufficient postMessage Validation



SNYK CODE | CWE-20 

SCORE
450

```
13320 |         var p = function() {
13321 |             e.contentDocument.body.appendChild(f);
13322 |             f.addEventListener("load", function() {
13323 |                 d.la = f.contentWindow;
13324 |                 e.contentWindow.addEventListener("message", function(q) {
```

The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

 gtmcontactus.js 

1 step in 1 file

Code Analysis

OverviewHistorySettings

L

Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

SNYK CODE

CWE-614

SCORE

450

9880

function Er(a, b, c, d) {

9881

var e = Ar(),

9882

f = window;

9883

wr(f) && (f.document.cookie = a);

Cookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

gtmcontactus.js

1 step in 1 file

Code Analysis

OverviewHistorySettings

L

Insufficient postMessage Validation

SNYK CODE

CWE-20

SCORE

450

13320

var p = function() {

13321

e.contentDocument.body.appendChild(f);

13322

f.addEventListener("load", function() {

13323

d.la = f.contentWindow;

13324

e.contentWindow.addEventListener("message", function(g) {

The origin of the received message is not checked. This means any site (even malicious) can send message to this window. If you don't expect this, consider checking the origin of sender.

gtmhomepage.js

1 step in 1 file

Learn about this type of vulnerability and how to fix it

Code Analysis

OverviewHistorySettings

L

Sensitive Cookie in HTTPS Session Without 'Secure' Attribute

SNYK CODE

CWE-614

SCORE

450

9880

function Er(a, b, c, d) {

9881

var e = Ar(),

9882

f = window;

9883

wr(f) && (f.document.cookie = a);

Cookie misses the Secure attribute (it is false by default). Set it to true to protect the cookie from man-in-the-middle attacks.

gtmhomepage.js

1 step in 1 file

Ignore across repository

View details

6.3 Key Vulnerabilities Discovered (Supply Chain Risk)

By targeting the GTM files (“gtmaboutus.js”, “gtmcontactus.js”, “gtmhomepag.js”), Snyk revealed critical vulnerabilities related to security configuration and data handling. These issues demonstrate weaknesses that an attacker could exploit to compromise user sessions or data.

Vulnerability	Severity (Snyk)	CWE ID	STRIDE Link	Relevance to BYKEA Case Study
Sensitive Cookie in HTTPS Session Without 'Secure' Attribute	Low	CWE-614	Information Disclosure	Configuration Failure: A session cookie being sent without the Secure attribute (even over HTTPS) leaves the cookie vulnerable if the user later accesses the site via insecure HTTP, risking session hijacking and unauthorized data access.
Insufficient postMessage Validation	Low	CWE-20	Tampering, Spoofing	Insecure Coding: The postMessage function is used without checking the origin of the received message. This means any malicious site could send a message to the BYKEA page, potentially forcing the application to execute unintended actions or leak data through a compromised iframe.

6.4 Linkage to Threat Model and Remediation

The findings from Snyk directly reinforce the **Security Misconfiguration** and **Weak Guardrails** themes of the original incident:

Original Threat/Finding	Mitigation Confirmed by Snyk Scan
Information Disclosure & Session Exposure	The "Sensitive Cookie" vulnerability confirms a failure to implement simple, standard cookie security controls (the Secure attribute).

Weak Developer/DevOps Guardrails	The "Insufficient postMessage Validation" demonstrates a failure in secure coding practices that modern SAST tools like Snyk are designed to catch <i>before</i> deployment. This is a critical deficiency in developer education and mandatory code review.
Continuous Oversight (SCA)	The overall process confirms that by integrating Snyk with GitHub, BYKEA can now perform Continuous Security Monitoring on all custom and third-party code, which is the necessary structural control to prevent future long-running exposures.