Techtweek Infotech
Service Excellence

WEB APPLICATION VAPT REPORT

techtweekinfotech.com

Vulnerability Assessment and Penetration Testing (VAPT) Report

Targets: admin.core.mysquegg.com
api.core.mysquegg.com
api.abbott.mysquegg.com

Assessor: Sahil Dubey Certified Information Systems Auditor® (CISA®)

Assessment Framework: OWASP Top 10 (2021)

Date: Aug 5, 2025

Version Control

Version	Date	Changes	Author
1.0	Aug 5, 2025	Initial draft	Sahil Dubey
1.1	Aug 5, 2025	Added detailed findings section	Sahil Dubey
1.2	Aug 4, 2025	Updated recommendations	Sahil Dubey
1.3	Aug 4, 2025	Presented to CTO	Saket Gunjan

Confidentiality Statement

This document contains sensitive information intended for authorised personnel only. Unauthorised distribution or use is strictly prohibited.

Access Control:

- Restrict access to the cybersecurity team and management
- Store securely in restricted-access drives or document management systems
- Distribute only via secure, encrypted channels

Disclaimer

A vulnerability assessment and penetration test (VAPT) is a real-time snapshot. The findings and recommendations in this report reflect the information gathered during the assessment period and do not account for any changes or modifications made outside that timeframe.

Due to the time-limited nature of this engagement, not all security controls may have been fully evaluated. The assessment was prioritised to identify the most exploitable weaknesses that an attacker could leverage. It is strongly recommended that similar assessments be conducted on at least an annual basis, by internal or third-party assessors, to ensure the continued effectiveness of security controls.

Assessment Overview

On Aug 4, 2025 admin.core.mysquegg.com, api.core.mysquegg.com, api.abbott.mysquegg.com and engaged Tecktweek Infotech to evaluate the security posture of its web application and supporting infrastructure. The assessment was conducted per industry best practices, including the OWASP Top 10 framework and relevant security testing methodologies.

All testing was performed externally, simulating an attacker's perspective with no internal access or prior knowledge. The assessment included both passive and active reconnaissance, vulnerability scanning, and manual verification of identified issues.

Phases of the VAPT engagement included:

- **Planning:** Defined assessment scope, objectives, and rules of engagement in collaboration with the client.
- **Discovery:** Conducted reconnaissance, scanning, and enumeration to identify potential vulnerabilities and weak points.
- **Attack:** Attempted to exploit identified vulnerabilities to confirm their existence and assess potential impact.
- Documented all findings, including confirmed vulnerabilities, failed
- **Reporting:** attempts, areas of strength, along with actionable remediation
- **recommendations**.

Tools and Environment

The following tools and platforms were used during the assessment:

- **Operating System:** Linux (Kali Linux 2024.1)
- **Directory and File Bruteforce:** ffuf, dirsearch, gobuster
- **Automated Vulnerability Scanning:** Nuclei (v3.4.4) with latest official and community templates
- **Injection Testing:** sqlmap (for SQL injection detection and exploitation)
- **DNS Enumeration:** nslookup
- **Technology Identification:** whatweb, whois (to fingerprint backend technologies and registrant details)
- **Port and Service Scanning:** nmap
- **Web Server Vulnerability Scanning:** Nikto (for outdated server versions and common misconfigurations)
- **Manual Testing and Verification:** Burp Suite Professional Edition, cURL, wget, browser-based testing, and manual XSS script testing

Executive Summary

This report presents the findings of a vulnerability assessment conducted on admin.core.mysquegg.com, api.core.mysquegg.com, api.abbott.mysquegg.com.The evaluation was performed using the OWASP Top 10 as the primary framework, which is the industry standard for identifying and prioritising the most critical web application security risks. The goal was to identify potential vulnerabilities, assess their impact, and provide recommendations for mitigation.

Findings Summary

1. Broken Access Control – Critical

A logic flaw in the `api.core.mysquegg.com /v2/profile` endpoint allows authenticated users to retrieve sensitive profile data of **all users**, bypassing access controls. This represents a **severe privacy risk** and violates OWASP A01:2021 – *Broken Access Control*.

2. Cryptographic Failures – Moderate

HTTPS and modern TLS configurations are enforced site-wide. However, **missing HSTS headers** on `admin.core.mysquegg.com` increase the potential for SSL stripping attacks in insecure network environments.

3. Injection – None Identified

No direct SQL, XSS, or command injection points were found during testing. While inputs appear safely handled, continued monitoring is advised to catch any contextual output encoding or trust violations in future code changes.

4. Insecure Design – High

Sensitive logic such as **admin verification checks** was discovered in frontend JavaScript (`index-9mWQGS6S.js`). This logic is **easily accessible**, allowing attackers to reverse-engineer and manipulate client-side workflows — a major concern per OWASP A04:2021 – *Insecure Design*.

5. Security Misconfiguration – Moderate

Several misconfigurations were noted:

Missing headers: `Strict-Transport-Security`, `Content-Security-Policy`, `Referrer-Policy`
Framework exposure: `X-Powered-By: Next.js`
Weak SPA behavior: Invalid routes return HTTP 200
Endpoint exposure: `/api/auth/providers` publicly lists available auth mechanisms

6. Vulnerable and Outdated Components – Low

The frontend discloses usage of **Next.js v15.2.2**. While no current CVEs were identified, exposing framework versions increases **future exploitability** risk.

7. Identification and Authentication Failures – Informational

While session cookies are secure, endpoints like `/v3/admin/check-admin` respond with permissive **CORS headers** and expose HTTP methods via unauthenticated **OPTIONS requests**. Additionally, `/api/auth/providers` leaks valid auth strategies.

8. Software and Data Integrity Failures – Moderate

Publicly accessible, unauthenticated access to files such as `.plist`, `.zip`, and `.mp4` on `admin.core.mysquegg.com` may allow for **tampering, reconnaissance, or leakage** of internal configuration or assets.

9. Security Logging and Monitoring Failures – High

Automated enumeration using invalid endpoint patterns (e.g., `/invalid1`–`/invalid20`) received **200 OK responses** with **no rate limiting or alerts**. This indicates the absence of effective **WAF, logging, or intrusion detection**, exposing the system to undetected reconnaissance attacks.

10. Server-Side Request Forgery (SSRF) – Not Exploitable

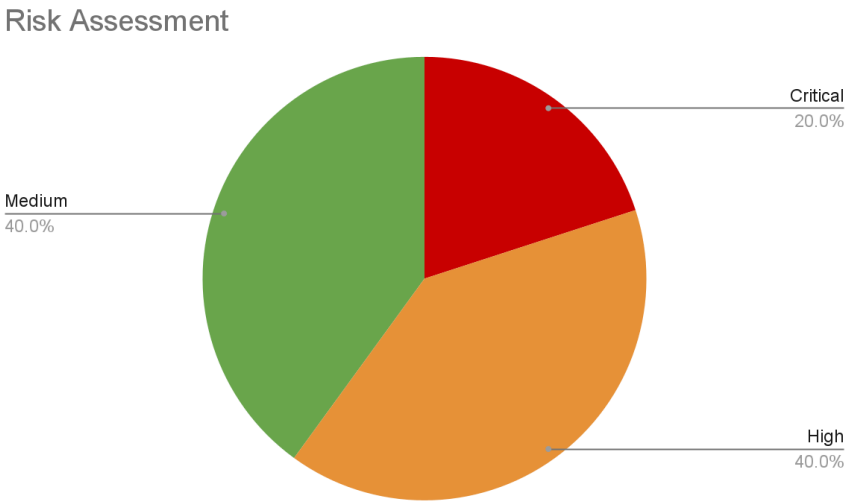
The `/api/fetch?url=` endpoint was tested, but **CloudFront effectively blocked POST/internal SSRF vectors**. No exploitable SSRF paths were found.

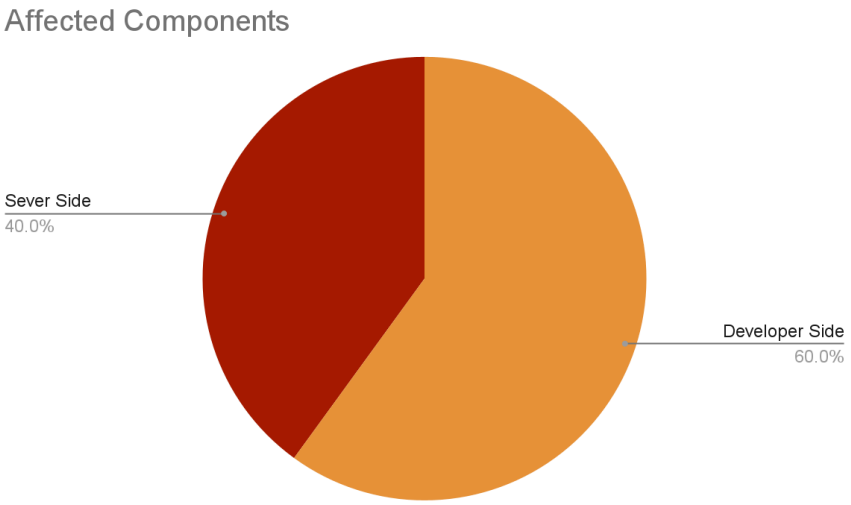
Finding Severity Ratings

The following table defines the severity levels and corresponding CVSS score ranges that are used throughout the document to assess vulnerability and risk impact.

Severity	CVSS V3 Score Range	Definition
Critical	9.0-10.0	Exploitation is straightforward and usually results in system-level compromise. It is advised to form a plan of action and patch immediately.
High	7.0-8.9	Exploitation is more challenging but can result in elevated privileges and potentially lead to data loss or downtime. It is advised to form a plan of action and patch as soon as possible.
Moderate	4.0-6.9	Vulnerabilities exist, but are not exploitable or require extra steps such as social engineering. It is advised to form a plan of action and patch after high-priority issues have been resolved.
Low	0.1-3.9	Vulnerabilities are non-exploitable but would reduce an organisation's attack surface. It is advised to form a plan of action and patch during the next maintenance window.
Informational	N/A	No vulnerability exists. Additional information is provided regarding items noticed during testing, strong controls, and additional documentation.

Findings Overview:





Vulnerability Status Table

OWASP Top 10 Category	admin.core.mysquegg.com	api.core.mysquegg.com	api.abbott.mysquegg.com
Broken Access Control	Passed	Failed	Passed
Cryptographic Failures	Passed	Passed	Passed
Injection	Passed	Passed	Passed
Insecure Design	Failed	Passed	Passed
Security Misconfiguration	Failed	Passed	Passed
Vulnerable and Outdated Components	Passed	Passed	Passed
Identification and Authentication Failures	Failed	Passed	Passed
Software and Data Integrity Failures	Passed	Passed	Passed
Security Logging and Monitoring Failures	Failed	Passed	Passed
Server-Side Request Forgery (SSRF)	Passed	Passed	Passed

Finding: Broken Access Control – Insufficient Authorization Checks in API Endpoint

Target: api.core.mysquegg.com
Status: Failed

What was discovered:

The API endpoint /v2/profile allows any logged-in user to access data belonging to other users without restriction.

During testing, an authenticated user (testuser13) was able to retrieve a complete list of other user profiles, including:

- Full names (e.g., “Bill Kenyon”, “Camila S”, “Danny Tung”)
- User IDs
- Friendship and request status (isFriend, isRequested, isPending)

This means there are no authorization checks in place to confirm whether the requesting user is permitted to view this information.

Why this is critical:

This is a textbook Broken Access Control issue — the #1 risk in the OWASP Top 10 (2023). Here's what this could mean for your business:

- Any user in your system can extract PII of all other users.
- If an attacker registers, they could scrape your entire user base, opening the door for:
    - Phishing attacks
    - Identity theft
    - Competitor intelligence

- You may be in violation of privacy laws like GDPR, HIPAA, or India's DPDP Act — exposing you to legal and financial penalties.
- This undermines user trust and could lead to brand damage if discovered.

Recommendations:

To mitigate this issue, the following changes are strongly recommended:

- Enforce authorization checks on the /v2/profile endpoint: return only the profile of the authenticated user.
- Implement Role-Based Access Control (RBAC) or Object-Level Access Control (OLAC) to ensure users can only access data they own or are permitted to view.
- Do not return full user lists unless explicitly authorized (e.g., admin).
- Log and alert on any abnormal access patterns or bulk profile queries.
- Apply rate limiting to this endpoint to reduce the risk of enumeration or scraping.

Proof of Concept:

Step 1: Authenticate and obtain bearer token

```
curl -X POST https://api.core.mysquegg.com/v2/auth/login \
  -H "Content-Type: application/json" \
  -d '{"username":"testuser13","password":"testuser13"}'
```

Step 2: Use token to access unauthorized data

```
curl -X GET https://api.core.mysquegg.com/v2/profile \
  -H "Authorization: Bearer <accessToken>"
```

Result:

The response includes multiple other user profiles with PII, such as:

- `{"id":3019, "name":"Bill Kenyon", "isFriend":false}`
- `{"id":15569, "name":"Camila S", "isFriend":false}`

This confirms that data isolation is missing and access control is broken.

```
┌──(root㉿kali)-[/home/rootrebel]
└─# curl -X GET https://api.core.mysquegg.com/v2/profile \
-H "Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJkYXRhIjp7Im5hbWUiOiJ0ZXN0dXNlcjEzIiwiZW1haWwiOiJ0ZXN0dXNlcjEzQGdtYWlsLmNvbSIsInVzZXJJZCI6MSwicm9sZUlkIjoxLCJ1c2VybmFtZSI6InRlc3R1c2VyMTMiLCJwcm9maWxlSWQiOjF9LCJpYXQiOjE3NTQyODUyMzYsImV4cCI6MTc1NDI4ODgzNn0.kIxsCCdtjq_Ki3tXzz8h71-0GPBuoKG5hL94QlmMYd8"

{"success":true,"data":[{"id":8319,"name":"Bigyaa Pokhrel","isFriend":false,"isRequested":false,"isPending":false},{"id":3019,"name":"Bill Kenyon","isFriend":false,"isRequested":false,"isPending":false},{"id":15569,"name":"Camila Sissan","isFriend":false,"isRequested":false,"isPending":false},{"id":13017,"name":"Danyela Hernández","isFriend":false,"isRequested":false,"isPending":false},{"id":495,"name":"Vallibus","isFriend":false,"isRequested":false,"isPending":false},{"id":11756,"name":"MICHELLE LAWSON","isFriend":false,"isRequested":false,"isPending":false},{"id":7581,"name":"kitty myers","isFriend":false,"isRequested":false,"isPending":false},{"id":6290,"name":"Deb Gestrich","isFriend":false,"isRequested":false,"isPending":false},{"id":8369,"name":"Pulkit Singh","isFriend":false,"isRequested":false,"isPending":false},{"id":3535,"name":"Tiden Belter","isFriend":false,"isRequested":false,"isPending":false},{"id":15570,"name":"jude thiele","isFriend":false,"isRequested":false,"isPending":false},{"id":4021,"name":"Rhyne Manlogon","isFriend":false,"isRequested":false,"isPending":false},{"id":499,"name":"Dau","isFriend":false,"isRequested":false,"isPending":false},{"id":10046,"name":"Prabin Clinic","isFriend":false,"isRequested":false,"isPending":false},{"id":3755,"name":"Ewan Sears","isFriend":false,"isRequested":false,"isPending":false},{"id":15571,"name":"Prabin Thakur","isFriend":false,"isRequested":false,"isPending":false},{"id":4024,"name":"Lorna Webb","isFriend":false,"isRequested":false,"isPending":false},{"id":3323,"name":"Cris","isFriend":false,"isRequested":false,"isPending":false},{"id":15264,"name":"Riminifisiocenter","isFriend":false,"isRequested":false,"isPending":false},{"id":4353,"name":"Madison","isFriend":false,"isRequested":false,"isPending":false},{"id":3528,"name":"Dawn","isFriend":false,"isRequested":false,"isPending":false},{"id":350,"name":"RichSherw","isFriend":false,"isRequested":false,"isPending":false},{"id":4202,"name":"Ash Jaffer","isFriend":false,"isRequested":false,"isPending":false},{"id":15572,"name":"aline dubsky","isFriend":false,"isRequested":false,"isPending":false},{"id":4355,"name":"Rachael O'Brien","isFriend":false,"isRequested":false,"isPending":false},{"id":15573,"name":"Abdelrahman Kaddoura","isFriend":false,"isRequested":false,"isPending":false},{"id":15574,"name":"Emily Alejandra Galiacho López","isFriend":false,"isRequested":false,"isPending":false},{"id":4973,"name":"Matej5281","isFriend":false,"isRequested":false,"isPending":false},{"id":15575,"name":"Elizabeth, Medina","isFriend":false,"isRequested":false,"isPending":false},{"id":4162,"name":"Zuleika","isFriend":false,"isRequested":false,"isPending":false},{"id":15576,"name":"Dan, Ramirez Morales","isFriend":false,"isRequested":false,"isPending":false},{"id":4992,"name":"Arjun","isFriend":false,"isRequested":false,"isPending":false},{"id":4752,"name":"Kevin Ridgeway","isFriend":false,"isRequested":false,"isPending":false},{"id":15577,"name":"Laura Patricia Magallanes","isFriend":false,"isRequested":false,"isPending":false},{"id":4624,"name":"R1sk 0L","isFriend":false,"isRequested":false,"isPending":false},{"id":137,"name":"Jim's Ego","isFriend":false,"isRequested":false,"isPending":false},{"id":15578,"name":"Diana, G","isFriend":false,"isRequested":false,"isPending":false},{"id":4760,"name":"Tony Fontanarosa","isFriend":false,"isRequested":false,"isPending":false},{"id":4316,"name":"Theo Armour","isFriend":false,"isRequested":false,"isPending":false},{"id":4761,"name":"Allan T","isFriend":false,"isRequested":false,"isPending":false},{"id":15579,"name":"Maria","isFriend":false,"isRequested":false,"isPending":false},{"id":4986,"name":"m sa","isFriend":false,"isRequested":false,"isPending":false},{"id":5020,"name":"Nancy Vail","isFriend":false,"isRequested":false,"isPending":false},{"id":15580,"name":"Anukoon K.","isFriend":false,"isRequested":false,"isPending":false},{"id":12386,"name":"Hilla M","isFriend":false,"isRequested":false,"isPending":false},{"id":4764,"name":"Laurel","isFriend":false,"isRequested":false,"isPending":false},{"id":15581,"name":"Karol Connors","isFriend":false,"isRequested":false,"isPending":false},{"id":5173,"name":"Sandra Navarro","isFriend":false,"isRequested":false,"isPending":false},{"id":3215,"name":"Lisa Kettler","isFriend":false,"isRequested":false,"isPending":false},{"id":15582,"name":"Marjorie Mendoza","isFriend":false,"isRequested":false,"isPending":false},{"id":5174,"name":"Macy Ticknor","isFriend":false,"isRequested":false,"isPending":false},{"id":15583,"name":"asad yuusuf","isFriend":false,"isRequested":false,"isPending":false},{"id":5302,"name":"Chase snarr","isFriend":false,"isRequested":false,"isPending":false},{"id":5326,"name":"Carlos F. Barcenas","isFriend":false,"isRequested":false,"isPending":false},{"id":238,"name":"Dipendra Shrestha","isFriend":false,"isRequested":false,"isPending":false},{"id":3164,"name":"danny tung","isFriend":false,"isRequested":false,"isPending":false},{"id":7892,"name":"David","isFriend":false,"isRequested":false,"isPending":false},{"id":2486,"name":"KEVIN SHANAHAN","isFriend":false,"isRequested":false,"isPending":false},{"id":15214,"name":"Kimmie","isFriend":false,"isRequested":false,"isPending":false},{"id":4371,"name":"星麗奈","isFriend":false,"isRequested":false,"isPen
```

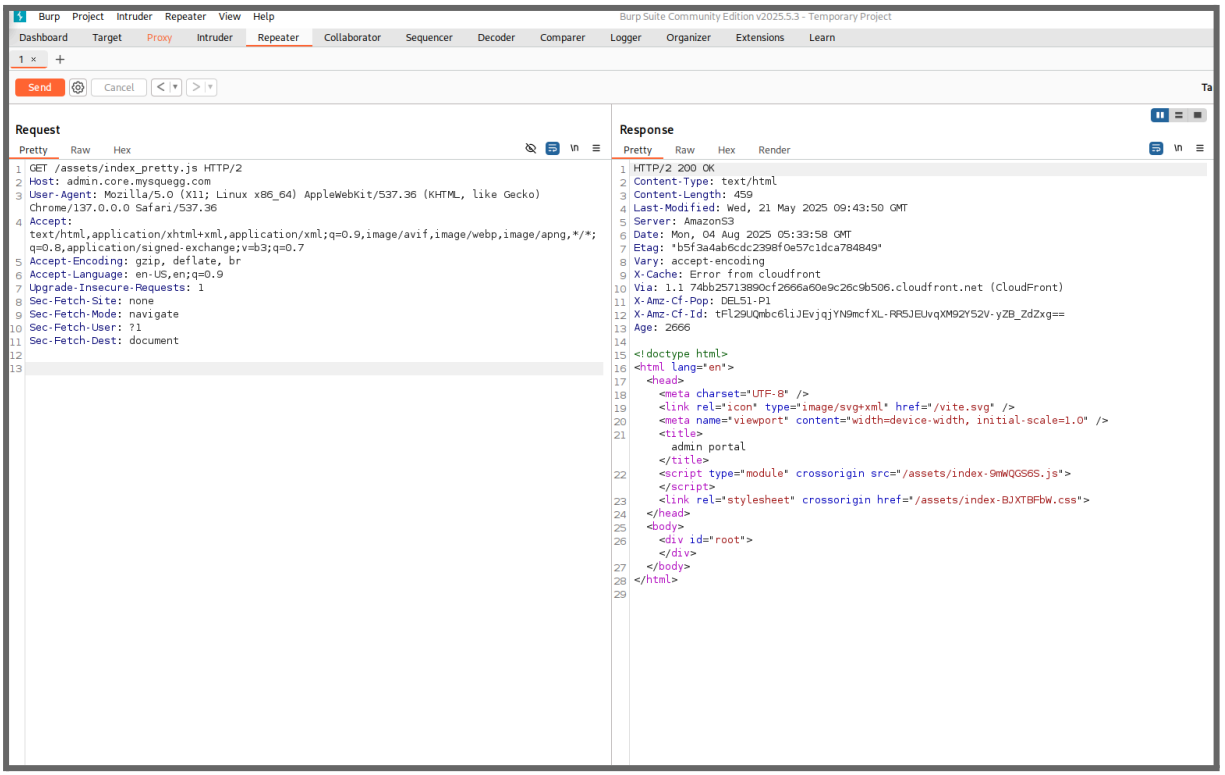
Finding: Insecure Design – Improper Handling of Invalid Resource Requests

Target: admin.core.mysquegg.com

Status: Failed

Description: The application is designed to return the main admin portal page with a 200 OK status code for requests to non-existent files or resources, rather than returning an appropriate error (such as 404 Not Found). This behavior was observed when requesting a missing JavaScript file (/assets/index_pretty.js), but instead of an error, the server responded with the main HTML page for the admin portal.

Proof of Concept:



Observed Response:

HTTP/2 200

content-type: text/html

...

<title>admin portal</title>

...

Expected Response:

A 404 Not Found error for missing or invalid resources.

Security Impact:
This design flaw allows attackers to probe for files and endpoints without receiving any indication of what is valid or invalid, making reconnaissance easier and increasing the risk of undetected attacks. Additionally, the lack of proper error handling and differentiation between valid and invalid requests can hinder security monitoring and incident response.

Recommendation:

- Update the application logic to return appropriate HTTP status codes (e.g., 404 Not Found) for requests to non-existent resources.
- Implement proper error handling and user feedback for invalid requests.
- Ensure that all unexpected or suspicious requests are logged and monitored for potential abuse.
- Review application design to ensure security controls are in place for handling errors and unexpected input.

Finding: Security Logging and Monitoring Failures – No Detection or Rate Limiting on Invalid Requests

Target: admin.core.mysquegg.com

Status: Failed

What was discovered:

During testing, we executed a simple automated script that sent 20 invalid HTTP requests to the application (`/invalid1` through `/invalid20`).

All requests:

- Returned HTTP 200 OK, even though the URLs didn't exist
- Were not rate-limited or blocked
- Triggered no signs of alerting or detection

This suggests the system lacks effective security monitoring, rate limiting, and anomaly detection mechanisms.

Why this is important:

Without basic detection and throttling, attackers can:

- Enumerate hidden endpoints
- Run brute-force or credential-stuffing attacks undetected
- Probe for exploitable behavior (e.g., upload bugs, admin paths)
- Bypass early warning systems that should flag this behavior

In simple terms, your system currently provides no visibility or defense against many forms of reconnaissance and automated attacks — giving attackers a free playground.

This maps to OWASP Top 10 A09:2021 – Security Logging and Monitoring Failures, and leaves the app vulnerable to stealth exploitation and delayed incident response.

Recommendations:

- Log all failed, invalid, or suspicious requests (e.g., 404s, malformed paths)
- Implement rate limiting (e.g., via CDN, WAF, or application-level logic)
- Block or delay responses for repeated suspicious patterns from the same IP
- Integrate with a SIEM or monitoring platform for alerting and threat detection
- Regularly review logs to identify brute-force attempts or endpoint fuzzing

Proof of Concept:

The following bash loop simulates automated endpoint fuzzing:

```
for i in {1..20}; do
  curl -s -o /dev/null -w "%{http_code} -> /invalid$i\n"
https://admin.core.mysquegg.com/invalid$i
done
```

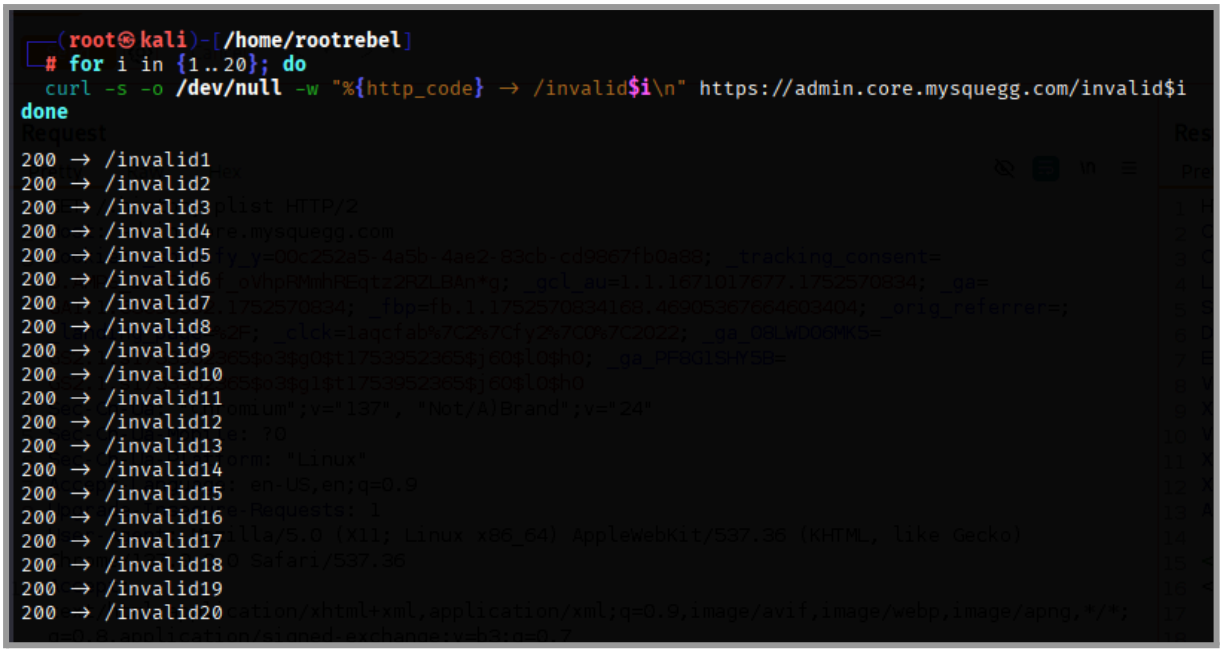
Observed Result:

200 -> /invalid1

200 -> /invalid2

...

200 -> /invalid20



Finding: Security Misconfiguration & Insecure Design – Improper Handling of Invalid and Sensitive File Requests

Target: admin.core.mysquegg.com
Status: Failed

Finding:

Automated directory and file fuzzing revealed that the server responds with 200 OK to requests for both non-existent and potentially sensitive files (such as .plist, .zip, .mp4, and hidden files like .git/HEAD), regardless of whether these files actually exist. The server does not differentiate between valid and invalid resources, and returns a generic response for all requests.

Impact:

This misconfiguration and insecure design allow attackers to:

- Enumerate files and directories without restriction, making it easier to discover hidden or sensitive resources.
- Perform reconnaissance and map the application's internal structure.

- Potentially identify endpoints or files that could be targeted for further exploitation.
- Bypass early warning systems, as the application does not provide proper error codes or logging for invalid or suspicious requests.

Recommendation:

- Configure the server to return appropriate HTTP status codes (e.g., 404 Not Found for non-existent files, 403 Forbidden for restricted resources).
- Restrict access to sensitive files and directories by enforcing proper authentication and authorization controls.
- Review and harden server configurations to prevent public access to internal, backup, or configuration files.
- Regularly audit exposed endpoints and file permissions, and implement monitoring to detect and alert on unauthorized access attempts or enumeration activity.

Proof of Concept:

The attached screenshot demonstrates the use of the ffuf tool to enumerate files and directories. All requests, including those for non-existent or sensitive files, receive a 200 OK response with identical content and size, confirming the lack of proper error handling and access controls.

```
┌──(root㉿kali)-[/home/rootrebel]
└─# ffuf -u https://admin.core.mysquegg.com/FUZZ -w /usr/share/wordlists/dirb/common.txt -e .plist,.zip,.mp4 -mc
200
        /'___\  /'___\           /'___\
       /\ \__/ /\ \__/  __  __  /\ \__/
       \ \ ,__\\ \ ,__\/\ \/\ \ \ \ ,__\
        \ \ \_/ \ \ \_/\ \ \_\ \ \ \ \_/
         \ \_\   \ \_\  \ \____/  \ \_\
          \/_/    \/_/   \/___/    \/_/

       v2.1.0-dev
________________________________________________

 :: Method           : GET
 :: URL              : https://admin.core.mysquegg.com/FUZZ
 :: Wordlist         : FUZZ: /usr/share/wordlists/dirb/common.txt
 :: Extensions       : .plist .zip .mp4
 :: Follow redirects : false
 :: Calibration      : false
 :: Timeout          : 10
 :: Threads          : 40
 :: Matcher          : Response status: 200
________________________________________________

.cache                  [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 847ms]
.cvsignore.plist        [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 843ms]
.git/HEAD               [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 837ms]
.cache.zip              [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 844ms]
.bashrc                 [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.bash_history.plist     [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 848ms]
.cache.plist            [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.git/HEAD.mp4           [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 843ms]
.history                [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 843ms]
.git/HEAD.plist         [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.forward                [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.cvs.zip                [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.forward.mp4            [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 838ms]
.cvs.mp4                [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 837ms]
.cvsignore              [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.bashrc.mp4             [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.history.mp4            [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.history.plist          [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 852ms]
                        [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.bashrc.zip             [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 864ms]
.bash_history.mp4       [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 847ms]
.history.zip            [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.bash_history.zip       [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.cvsignore.mp4          [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 849ms]
.forward.zip            [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 847ms]
.config.plist           [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 845ms]
.bashrc.plist           [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 843ms]
.cvs                    [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 849ms]
.zip                    [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 843ms]
.mp4                    [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 846ms]
.config.zip             [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 847ms]
.config.mp4             [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 844ms]
.bash_history           [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 859ms]
.listings.mp4           [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 316ms]
.listing                [Status: 200, Size: 459, Words: 57, Lines: 15, Duration: 331ms]
```

Finding: Identification and Authentication Failures – Permissive CORS and Unauthenticated Preflight on Sensitive Endpoint

Target: admin.core.mysquegg.com
Status: Failed

Finding:

The `/v3/admin/check-admin` endpoint responds to unauthenticated preflight (OPTIONS) requests with permissive CORS headers, allowing any origin (`Access-Control-Allow-Origin: *`) and multiple HTTP methods, including GET, HEAD, PUT, PATCH, POST, and DELETE. As shown in the screenshot, the server responds with a 204 No Content status and does not require authentication for this preflight request. This behavior exposes internal functionality and increases the authentication attack surface.

Impact:

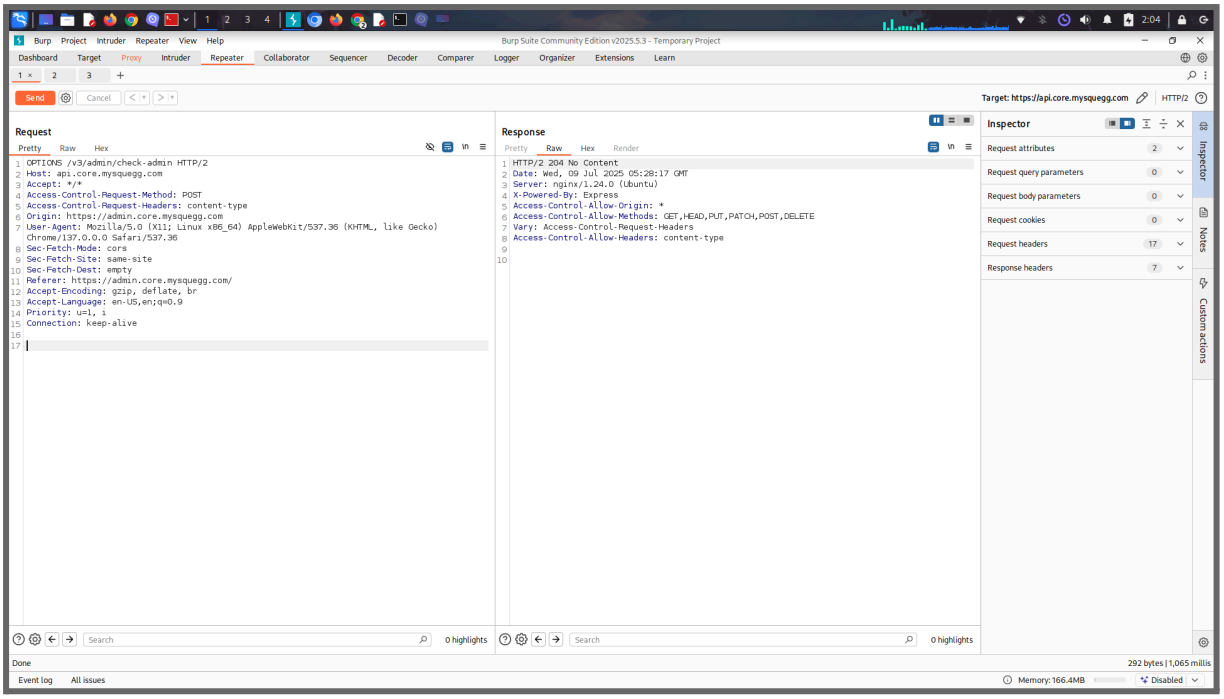
Permissive CORS policies and unauthenticated preflight responses can be exploited by attackers to probe internal APIs, enumerate available methods, and potentially bypass authentication controls. This increases the risk of cross-origin attacks and unauthorized access to sensitive endpoints.

Recommendation:

Restrict CORS policies to only allow trusted origins and limit allowed HTTP methods to those necessary for application functionality. Require authentication for all sensitive endpoints, including preflight requests, and avoid exposing internal API capabilities to unauthenticated users.

Proof of Concept:

The following screenshot demonstrates an unauthenticated OPTIONS request to the `/v3/admin/check-admin` endpoint, which returns permissive CORS headers and exposes available HTTP methods:



Finding: No Exploitable Vulnerabilities Identified During Testing

Target: api.abbott.mysquegg.com

Status: Passed

Summary:

The API was tested for common OWASP vulnerabilities and misconfigurations. No exploitable issues were identified during the engagement.

Tests Performed:

- Header analysis (security headers, CSP, HSTS, CORS)
- Endpoint fuzzing using ffuf, dirsearch
- CORS abuse tests using crafted Origin headers
- Auth probing with curl, Postman
- Automated scanning with wapiti, nmap, and nuclei

Notable Security Controls Identified:

- Strong Content Security Policy (CSP)
- Proper use of HTTPS with HSTS
- Minimal exposed endpoints
- No version leaks or server-side error disclosures

Observations:

- CORS is configured with Access-Control-Allow-Origin: *, but no sensitive data was accessible.
- X-XSS-Protection header is disabled (0), which may affect legacy browser users.

Recommendation:

- Monitor and harden CORS configuration if new endpoints are added in future.
- Consider enabling X-XSS-Protection: 1; mode=block for additional browser-level security.

Conclusion:

The security assessment revealed critical gaps in both access control and monitoring. Sensitive user data was exposed through the `/v2/profile` API due to missing authorization checks, allowing any authenticated user to view other users' PII. Additionally, the lack of rate limiting and monitoring enabled automated requests to go undetected. These issues significantly increase the risk of data breaches and regulatory non-compliance, and should be addressed with priority to strengthen the platform's overall security posture.