TN7

Penetration Testing Report

1st August 2025

FAILSAFE

Table of Contents

Project Details

Project	TN7 Penetration Testing
Web URL	https://mint.tn7.co/
API URL	https://wfstudiobe.morpheuslabs.io/
Timeline	Initial Report - 10th July 2025
	Final Report - 1st August 2025

Structure & Organization of The Security Report

Issues are tagged as “Open”, “Acknowledged”, “Partially Resolved”, “Resolved” or “Closed” depending on whether they have been fixed or addressed.

- Open: The issue has been reported and is awaiting remediation from developer team.
- Acknowledged: The developer team has reviewed and accepted the issue but has decided not to fix it.
- Partially Resolved: Mitigations have been applied, yet some risks or gaps still remain.
- Resolved: The issue has been fully addressed and no further work is necessary.
- Closed: The issue is deemed no longer pertinent or actionable.

Furthermore, the severity of each issue is written as assessed by the risk of exploitation or other unexpected or otherwise unsafe behavior:

Critical	The issue affects the application in such a way that funds may be lost, allocated incorrectly, or otherwise result in a significant loss.
High	The issue affects the ability of the application to compile or operate in a significant way.
Medium	The issue affects the ability of the application to operate in a way that doesn’t significantly hinder its behavior.
Low	The issue has minimal impact on the application’s ability to operate.
Info	The issue is informational in nature and does not pose any direct risk to the application’s operation.

Project Goals

1. Identify and document security vulnerabilities in TN7 platform's authentication and mint flows.
2. Assess the effectiveness of existing security controls and recommend improvements.
3. Evaluate the platform's resilience against common web application attacks (e.g., race conditions, CSRF, XSS, brute force, enumeration).
4. Provide actionable remediation guidance to improve the overall security posture.

Methodology

Testing Approach

The testing approach combines traditional web application security testing frameworks with blockchain-specific threat modeling to provide comprehensive coverage of both conventional and emerging attack vectors.

Industry Standard Frameworks

The assessment methodology is based on established industry frameworks including:

- **OWASP Testing Guide v4.0** - Comprehensive web application security testing methodology
- **NIST SP 800-115** - Technical guide to information security testing and assessment
- **PTES (Penetration Testing Execution Standard)** - Structured approach covering pre-engagement through reporting

Backend dApp Security Testing Methodology

Backend decentralized applications present unique security challenges that require specialized testing approaches beyond traditional web application assessments. Our methodology addresses key areas including:

API Security Assessment: Comprehensive evaluation of REST/GraphQL endpoints, authentication mechanisms, authorization controls, and data validation processes. This includes testing for common vulnerabilities such as injection attacks, broken authentication, sensitive data exposure, and insufficient logging.

Testing Phases

The penetration test follows a structured approach consisting of five distinct phases:

1. **Reconnaissance & Information Gathering** - Passive and active information collection about the target infrastructure, technology stack, and potential attack surfaces
2. **Vulnerability Assessment** - Systematic identification of security weaknesses using automated tools and manual analysis techniques

3. **Exploitation & Proof of Concept** - Validation of identified vulnerabilities through controlled exploitation attempts with documented proof-of-concept demonstrations
4. **Post-Exploitation Analysis** - Assessment of potential impact, privilege escalation possibilities, and lateral movement opportunities
5. **Reporting & Remediation Guidance** - Documentation of findings with detailed remediation recommendations and risk prioritization

In-scope

- Backend: https://wfstudiobe.morpheuslabs.io/
- Staging Frontend: https://viu-tn7-nft.dev.morpheuslabs.io/
- Production Frontend (Deployed fixes): https://mint.tn7.co/
- Environment: Staging/Production
- Intrusive Tests: Yes
- Type of Test: Blackbox Testing

Summary of Findings

Severity	Total	Open	Acknowledged	Partially Resolved	Resolved	Closed
Critical	2	-	-	-	2	-
High	3	-	-	-	3	-
Medium	4	-	1	1	2	-
Low	2	-	1	-	1	-
Info	-	-	-	-	-	-
Total	11	0	2	1	8	0

#	Findings	Severity	Status
1	Price Manipulation to Any Amount	Critical	Resolved
2	Static Nonce and Missing Domain Binding in Authentication	Critical	Resolved
3	API Key Validation Bypass in Stripe Webhook	High	Resolved
4	Wallet Mismatch And Authorization Bypass	High	Resolved
5	Authorization Token Remains Valid After Logout or New Login	High	Resolved
6	No Sanitization of Malicious Inputs May Lead to Potential SQL Injection	Medium	Resolved
7	Missing Security Headers in HTTP Response	Medium	Partially Resolved
8	General Patch Management	Medium	Acknowledged
9	Lack of Rate Limiting and Brute Force Protection	Medium	Resolved
10	Always Returning HTTP 200 on Errors	Low	Acknowledged
11	Hard Coded API Key Disclosure	Low	Resolved

Finding 1: Price Manipulation to Any Amount

Severity: Critical

Status: Resolved

Description:

No server-side enforcement of pricing; client-controlled amount field is trusted without validation. Attackers can manipulate the amount to any arbitrary value, allowing purchase of high-value NFTs at significantly reduced or even zero cost.

Request with “amount”:“1” for “ExpensiveNFT” returned success:true and created a Stripe intent for 417 ¢ (test currency rounding), demonstrating complete lack of price validation.

Impact:

1. Direct revenue loss; attackers can mint NFTs at any manipulated price.
2. Complete undermining of marketplace pricing integrity and economic model.
3. Potential for massive financial losses through systematic price manipulation.

Proof of Concept:

```
{"name":"ExpensiveNFT","amount":"1"}
```

Server returns client_secret for under-priced intent

Remediation:

Implement server-side price validation:

1. Server must look up NFT price by blockchain and ignore client-supplied amount.
2. Implement signed price tokens or obtain pricing exclusively from backend DB.
3. Add minimum price validation and maximum price limits.
4. Implement price verification against blockchain data.

Finding 2: Static Nonce and Missing Domain Binding in Authentication

Severity: Critical

Status: Resolved

Source: Frontend Authentication flow, API Webhook endpoints on wfstudiobe.morpheuslabs.io

Description:

The authentication process uses static nonces per wallet address and lacks domain binding in signature messages. This enables attackers to create phishing sites that prompt users to sign the same message format used by the legitimate platform, then use the signed message and wallet address to authenticate to the morpheuslabs platform. The backend API accepts signatures without enforcing unique challenges or domain validation.

Both the frontend and backend are vulnerable because the authentication system uses static nonces that never change for each wallet address. The signature message "You are logging into the system with the nonce: 135107" lacks domain binding and can be signed on any malicious site. An attacker can create a phishing site, ask the victim to sign this message format, and then use the signed message and wallet address to authenticate as the victim on the real platform. There is no server-side challenge uniqueness or domain validation to ensure the message is tied to the legitimate authentication session.

```
Authentication Request:
{
  "data": {
    "message": "You are logging into the system with the nonce: 135107"
  }
}
```

Impact:

1. Attackers can create phishing sites that trick users into signing the same message format used by the legitimate platform.
2. Signed messages can be replayed for up to 24 hours to gain unauthorized access to user accounts.
3. No domain binding ensures users may sign messages on malicious sites without realizing they're authenticating to the real platform.
4. Static nonces allow signature reuse and replay attacks.

Proof of Concept:

1. Create a phishing site that asks the victim to sign the message "You are logging into the system with the nonce: 135107"

2. Send the following to the real platform API:

```
POST /api/v1/webhooks/{endpoint}
{
  "wallet_address": "<victim_wallet>",
  "signature": "<victim_signature>"
}
```

3. If successful, the attacker receives a JWT token and is authenticated as the victim for 24 hours.

Remediation:

Implement secure authentication with proper challenge-response mechanism:

1. Server-Generated Nonces: Implement server-generated, unique nonces for each authentication attempt, rolling forward per wallet address.
2. Domain Binding: Include domain/URL in signature message to prevent cross-site signature harvesting.
3. Timestamp Validation: Add timestamp validation with short expiration periods.
4. Wallet Address Binding: Include wallet address in signature message for additional binding.

Finding 3: API Key Validation Bypass in Stripe Webhook

Severity: High

Status: Resolved

Description:

The webhook processes requests even when the x-api-key header is wrong or completely absent. Authentication logic trusts a hard-coded placeholder key (0ZJZGuNVaF0…) and does not reject invalid or missing keys, allowing unauthenticated traffic.

Requests containing an obviously invalid key (“x-api-key”: “invalid-api-key”) or no key at all received HTTP 200 with a full Stripe payment_intent object.

Impact:

1. Anyone can hit the webhook and create/confirm payment intents.
2. Enables automated fraud, data poisoning and potential DoS without rate limits.

Proof of Concept:

```
curl -X POST https://wfstudiobe.morpheuslabs.io/api/v1/webhooks/XYZ \
-H "Content-Type: application/json" \
-H "x-api-key: invalid-api-key" \
-d '{"walletAddress":"0…x","name":"Test","amount":"1"}'
#  HTTP 200, success:true
```

Remediation:

Implement proper API key validation:

1. Verify API keys server-side against a trusted store.
2. Reject missing/unknown keys with 401 Unauthorized.
3. Rotate keys and log failed attempts.
4. Implement rate limiting on webhook endpoints.

Finding 4: Wallet Mismatch And Authorization Bypass

Severity: High

Status: Resolved

Description:

The wallet address in the JSON body is never cross-checked against the wallet embedded in the JWT. Attackers can present any arbitrary wallet (e.g., 0x1234567890abcdef1234567890abcdef12345678) while re-using a valid JWT belonging to a different address.

Payloads with mismatched or even syntactically invalid addresses were accepted and processed with HTTP 200.

Impact:

1. Complete loss of per-wallet authorization; payments can be executed on behalf of other users.
2. Breaks audit trails and financial accountability.

Proof of Concept:

```
{
  "walletAddress": "0x1234567890abcdef1234567890abcdef12345678",
  "name": "EvilUser",
  "amount": "1"
}
```

JWT still carries victim's real wallet - server responds 200.

Remediation:

Implement proper wallet authorization:

1. Extract wallet address from JWT claims and compare it to walletAddress in the body.
2. Reject on mismatch and validate checksum (EIP-55).
3. Implement proper session validation for each request.
4. Add audit logging for all payment operations.

Finding 5: Authorization Token Remains Valid After Logout or New Login

Severity: High

Status: Resolved

Source: Frontend Authentication flow, API Webhook endpoints on wfstudiobe.morpheuslabs.io

Description:

Although the authorization token is deleted from localStorage on logout or when a new login occurs, the backend does not invalidate the old token. This means the token can still be used to call protected API endpoints even after the user has logged out or logged in with a different wallet.

Currently the JWT is expired after 24 hours, but revocation is still needed, since after logout old JWTs should be rejected.

On logout or wallet disconnect, the frontend removes the token from localStorage.

However, the backend does not revoke the token, so it remains valid until it naturally expires (or indefinitely, if no expiry is set).

If an attacker or malicious script obtains the token before logout, it can continue to use the token to access protected APIs, even after the user logs out or logs in with a new wallet.

Impact:

1. Old tokens can be used to access protected resources after logout or wallet change.
2. If a token is stolen, it can be used for unauthorized access until it expires.
3. Users may believe they are logged out, but their session remains valid on the backend.

Proof of Concept:

1. Connect wallet and obtain a token.
2. Connect wallet again with the same wallet (token is removed from localStorage).
3. Use the old token in an API request; the backend still accepts it.

Remediation:

Implement proper token revocation and session management:

1. Token Revocation: Implement token revocation or blacklisting on the backend when a user logs out or logs in with a new wallet.
2. Short-Lived Tokens: Use short-lived tokens and require re-authentication frequently.

3. Refresh Token Logic: Consider using refresh tokens with proper invalidation logic.
4. Session Tracking: Maintain a server-side session registry to track active tokens.
5. Immediate Invalidation: Immediately invalidate tokens upon logout or new authentication.

Auditor Response:

Developer acknowledges issue and have resolved the issue by implementing a shorter session timeout. However, issue remains partially resolved as token refresh mechanism and session revocation on logout is not implemented.

Finding 6: No Sanitization of Malicious Inputs May Lead to Potential SQL Injection

Severity: Medium

Status: Resolved

Description:

User-supplied metadata.customer_name is inserted into backend without sanitization. Payload metadata.customer_name": "'; DROP TABLE users; –" was accepted and echoed back, indicating absence of input sanitization or parameterized DB access.

Server responded HTTP 200 with the malicious string intact in the returned payment_intent.

Impact:

1. Potential execution of destructive SQL commands.
2. Data loss, credential leakage, or full DB compromise.

Proof of Concept:

```
{"metadata":{"customer_name":"'; DROP TABLE users; --"}}
```

Remediation:

Implement proper input validation and sanitization:

1. Switch to parameterized queries / ORM.
2. Validate and escape all metadata fields.
3. Apply allow-list on acceptable characters.
4. Implement input length limits and character restrictions.

Finding 7: Missing Security Headers in HTTP Response

Severity: Medium

Status: Partially Resolved

Description:

The application lacks essential security headers and content security policy implementation. This includes:

1. Clickjacking Vulnerability: The application can be embedded in iframes, allowing attackers to overlay deceptive elements over the legitimate interface. Users may unknowingly click on hidden elements controlled by attackers.
2. Missing Content Security Policy: No CSP headers are implemented, which means the application cannot control which sources of content (scripts, styles, images) are allowed to load. This increases vulnerability to cross-site scripting (XSS), clickjacking, and code injection attacks.
3. Missing Security Headers: The application lacks critical security headers including X-Content-Type-Options and Strict-Transport-Security (HSTS). Without X-Content-Type-Options, browsers may perform MIME sniffing and misinterpret file types. Without HSTS, the site is exposed to man-in-the-middle attacks where HTTPS connections can be downgraded to insecure HTTP.

Impact:

These security misconfigurations create multiple attack vectors:

1. Attackers can trick users into performing sensitive actions through clickjacking, such as making unauthorized transactions or revealing personal information.
2. The absence of CSP allows malicious scripts to execute, potentially leading to data theft and unauthorized user actions.
3. Missing security headers expose users to MIME confusion attacks and man-in-the-middle attacks, risking sensitive data interception.

Proof of Concept:

Frameable Response Test:

1. Create a file with html extension and save the following URL embedded in an iframe: https://viu-tn7-nft.dev.morpheuslabs.io/
2. Open the file in the browser and observe the application UI embedded.

Content Security Policy Test:

1. In Burp Suite, go to the HTTP history under the Proxy tab.
2. Select any response from the application (e.g., the main page or authenticated page).
3. Analyze the response headers for the presence of the Content-Security-Policy header.
4. Observe that no CSP header is present, indicating that no content security policy is enforced.

Security Headers Test:

1. Clone the shcheck tool from github and run the following command:
2. `python shcheck.py https://viu-tn7-nft.dev.morpheuslabs.io -d`

Remediation:

Implement comprehensive security headers and content security policy:

1. Add the following headers to all HTTP responses:

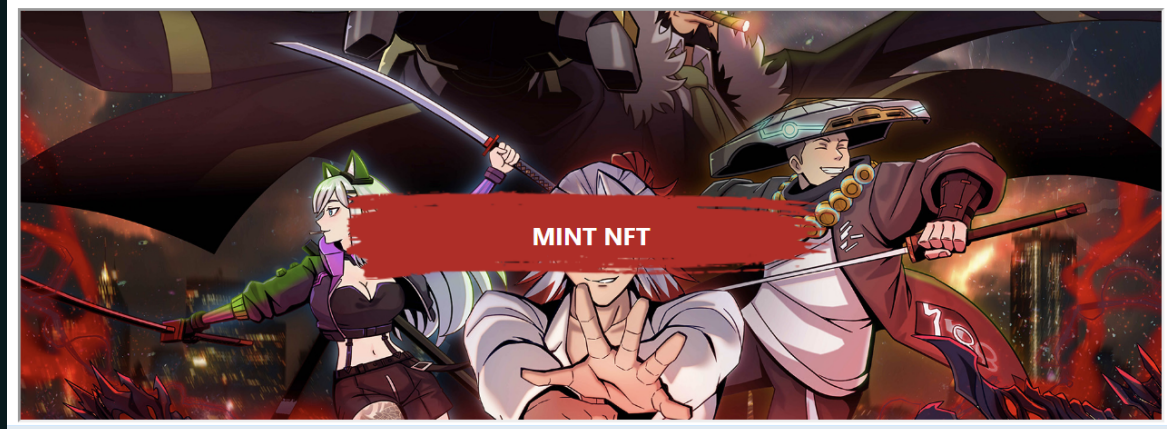
```
X-Frame-Options: DENY
X-Content-Type-Options: nosniff
Referrer-Policy: strict-origin-when-cross-origin
Content-Security-Policy: default-src 'self' (adjust as needed)
Permissions-Policy: geolocation=(), microphone=(), camera=()
X-XSS-Protection: 1; mode=block
```

2. Review and restrict `access-control-allow-origin` to only trusted origins for authenticated or sensitive endpoints.

References: - OWASP Security Misconfiguration

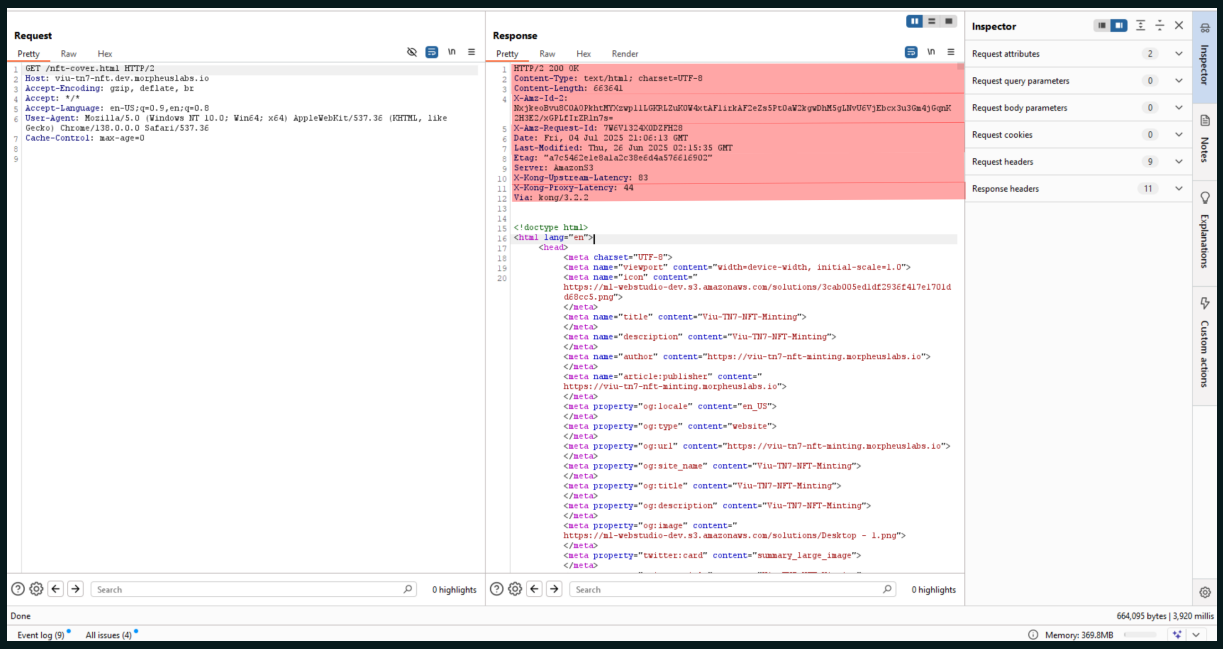
Attachments:

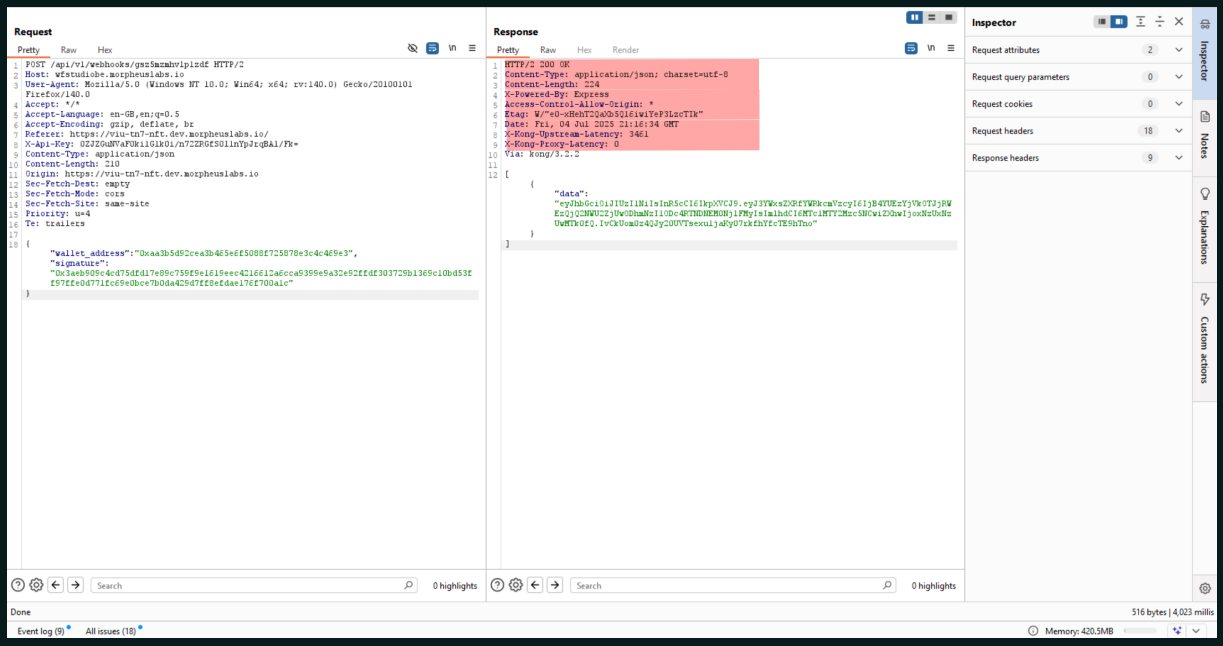
Clickjacking on https://viu-tn7-nft.dev.morpheuslabs.io/



Console Elements Sources Memory Performance Application Network Lighthouse Recorder

```
<!DOCTYPE html>
<html> scroll
  <head></head>
... <body> == $0
    <h1>Clickjacking on https://viu-tn7-nft.dev.morpheuslabs.io/</h1>
    <iframe src="https://viu-tn7-nft.dev.morpheuslabs.io/" style="width: 50%; height:300px;">...</iframe>
  </body>
</html>
```



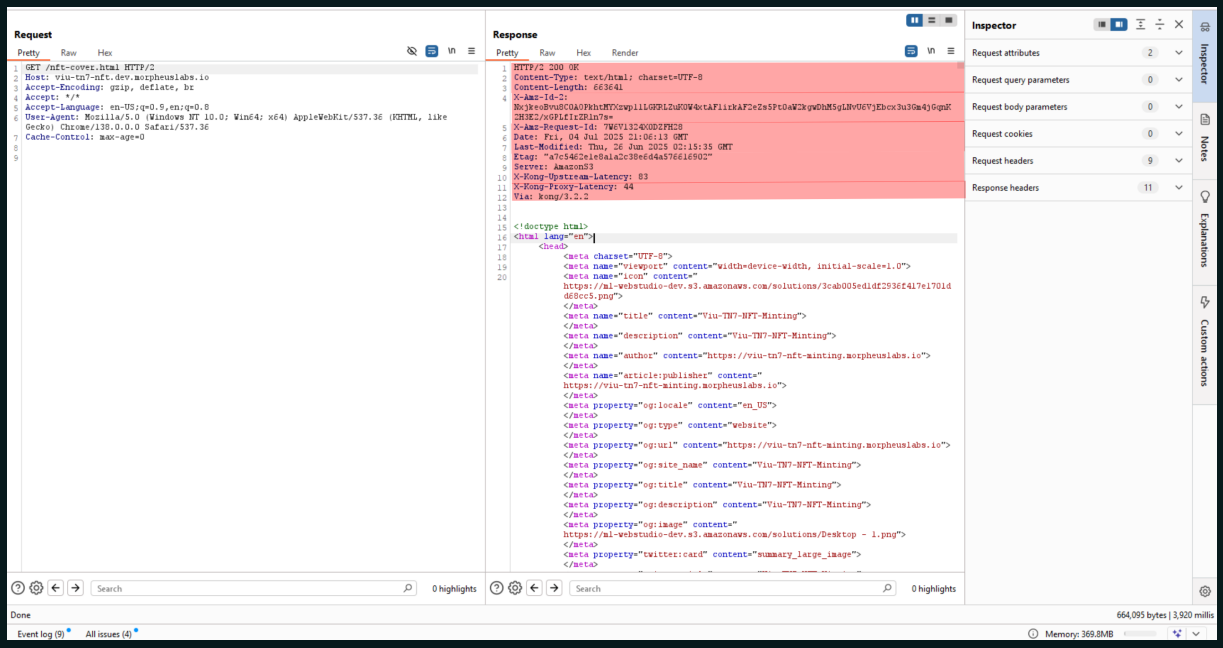



```
┌──(kali㉿kali)-[~/tools/shcheck]
└─$ python shcheck.py https://wfstudiobe.morpheuslabs.io -d

 > shcheck.py - santoru ..........................

 Simple tool to check security headers on a webserver

[*] Analyzing headers of https://wfstudiobe.morpheuslabs.io
[!] URL Returned an HTTP error: 404
[*] Effective URL: https://wfstudiobe.morpheuslabs.io
[!] Missing security header: X-Content-Type-Options
[!] Missing security header: Strict-Transport-Security
[!] Headers analyzed for https://wfstudiobe.morpheuslabs.io
[+] There are 0 security headers
[-] There are not 2 security headers
```




```
┌──(kali㉿kali)-[~/tools/shcheck]
└─$ python shcheck.py https://viu-tn7-nft.dev.morpheuslabs.io/nft-cover.html -d

 > shcheck.py - santoru ..............................

 Simple tool to check security headers on a webserver

[*] Analyzing headers of https://viu-tn7-nft.dev.morpheuslabs.io/nft-cover.html
[*] Effective URL: https://viu-tn7-nft.dev.morpheuslabs.io/nft-cover.html
[!] Missing security header: X-Content-Type-Options
[!] Missing security header: Strict-Transport-Security

[!] Headers analyzed for https://viu-tn7-nft.dev.morpheuslabs.io/nft-cover.html
[+] There are 0 security headers
[-] There are not 2 security headers
```

Auditor Response:

Developer has implemented security headers for the production mint.tn7.co website. However, CSP is not configured yet for the API endpoint wfstudiobe.morpheuslabs.io. There is also a CSP misconfiguration: The inclusion of ‘unsafe-inline’ and ‘unsafe-eval’ in both script-src and style-src directives significantly weakens the policy, allowing inline scripts and dynamic code execution, which are the very vectors CSP aims to block.

Finding 8: General Patch Management

Severity: Medium

Status: Acknowledged

Description:

The application uses outdated software components that contain known vulnerabilities. The Kong gateway server is running version 3.2.2, which has a documented CVE-2023-44487 vulnerability that allows denial of service attacks through HTTP/2 protocol exploitation.

Software and Installed Version: - kong/3.2.2

Current vulnerabilities:

- CVE-2023-44487:
  - Kong Server Uncontrolled Resource Consumption Vulnerability
  - The HTTP/2 protocol allows a denial of service (server resource consumption) because request cancellation can reset many streams.
  - http://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2023-44487
- Link to latest version:
  - https://developer.konghq.com/gateway/changelog/#3-11-0-0

Impact:

Outdated software components pose significant security and stability risks. The identified vulnerability in Kong 3.2.2 could allow attackers to consume server resources and cause service disruptions. Continued use of unpatched software increases the likelihood of exploitation and system issues.

Code:

```
Case 1. (viu-tn7-nft.dev.morpheuslabs.io):
Request
GET /nft-cover.html HTTP/2
Host: viu-tn7-nft.dev.morpheuslabs.io
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML
    , like Gecko) Chrome/138.0.0.0 Safari/537.36
Cache-Control: max-age=0
Response
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
```

```
Content-Length: 663641
X-Amz-Id-2:
NxjkeoBvu8COAOPkhtMYXzwp11LGKRLZuKOW4xtAF1irkAF2eZs5Pt0aW2kgwDhM5gLNvU6Vj
  Ebcx3u3Gm4jGqnK2H3E2/xGPLfIrZRln7s=
X-Amz-Request-Id: 7W6V1324X0DZFH28
Date: Fri, 04 Jul 2025 21:06:13 GMT
Last-Modified: Thu, 26 Jun 2025 02:15:35 GMT
Etag: "a7c5462e1e8a1a2c38e6d4a576616902"
Server: AmazonS3
X-Kong-Upstream-Latency: 83
X-Kong-Proxy-Latency: 44
Via: kong/3.2.2
<!doctype html><html lang="en"><head>...</script></html>

Case 2. (wfstudiobe.morpheuslabs.io):
Request
POST /api/v1/webhooks/gsz5mzmhvlp1zdf HTTP/2
Host: wfstudiobe.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101
    Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://viu-tn7-nft.dev.morpheuslabs.io/
X-Api-Key: 0ZJZGuNVaF0ki1G1k0i/n72ZRGfS0l1nYpJrqBA1/Fk=
Content-Type: application/json
Content-Length: 210
Origin: https://viu-tn7-nft.dev.morpheuslabs.io
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Priority: u=4
Te: trailers
{"wallet_address":"","signature":""}
Response
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 224
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"e0-e/Rr+cRw6/Tmp+9+G6Hp5GxV8cM"
Date: Fri, 04 Jul 2025 21:25:57 GMT
X-Kong-Upstream-Latency: 2023
X-Kong-Proxy-Latency: 0
Via: kong/3.2.2
[{"data":""}]
```

Remediation:

Implement a comprehensive patch management strategy:

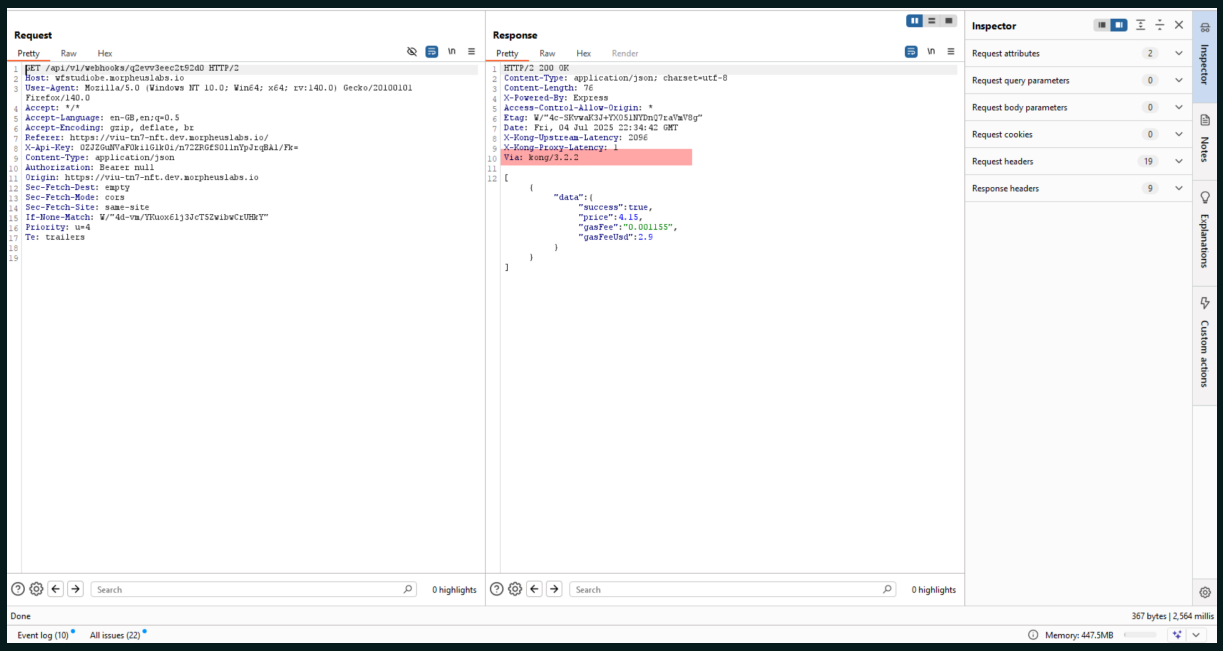
1. Update Kong Gateway: Upgrade to the latest stable version (3.11.0 or newer) to address the CVE-2023-44487 vulnerability.
2. Establish Patch Management Policy: Create formal procedures for identifying, testing,

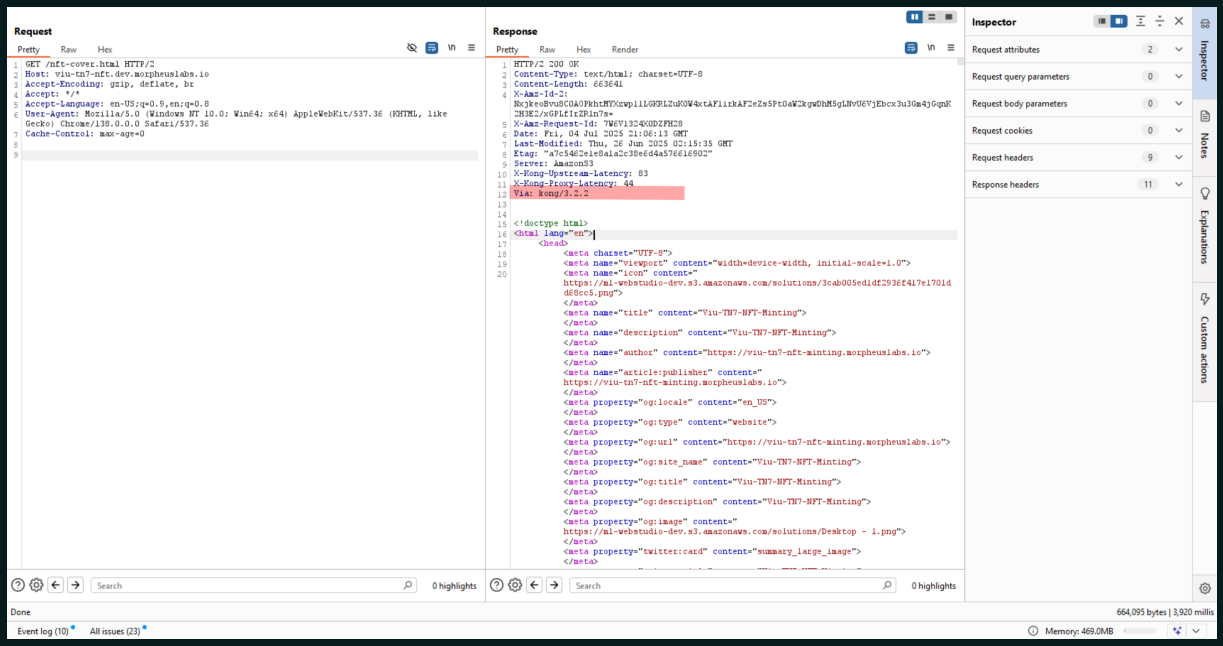
and deploying security patches.

3. Automated Vulnerability Scanning: Implement tools to regularly scan for missing patches and assess associated risks.
4. Testing Environment: Test patches in a controlled staging environment before production deployment.
5. Backup and Rollback Plans: Maintain system backups and rollback procedures in case patches cause instability.
6. Note: Due to the dynamic nature of security vulnerabilities, always check for the latest releases and patches rather than relying solely on the versions listed in this report.

References: - OWASP Vulnerable and Outdated Components

Attachments:




Request
Pretty Raw Hex
GET /nft-cover.html HTTP/2
Host: viu-tn7-nft.dev.morpheuslabs.io
Accept-Encoding: gzip, deflate, br
Accept: */*
Accept-Language: en-US;q=0.9,en;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/138.0.0.0 Safari/537.36
Cache-Control: max-age=0
Response
Pretty Raw Hex Render
HTTP/2 200 OK
Content-Type: text/html; charset=UTF-8
Content-Length: 663641
X-Amz-Id-2:
NxjkeoBvu8COAOFkhtMYXzwp1lLGKRLZuK0W4ktAFiirkAF2eZs5PtOaWZkgwDhM5gLNvU6VjEbcx3u3Gm4jGqnK2H3E2/xGPLfIrZRln7s=
X-Amz-Request-Id: 7WEV1324X0DZPH28
Date: Fri, 04 Jul 2025 21:06:13 GMT
Last-Modified: Thu, 26 Jun 2025 02:15:35 GMT
Etag: "a7c54626e1e8ala2c38e6d4a576616902"
Server: AmazonS3
X-Kong-Upstream-Latency: 83
X-Kong-Proxy-Latency: 44
Via: kong/3.2.2
<!doctype html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<meta name="icon" content="
https://ml-webstudio-dev.s3.amazonaws.com/solutions/3cab005edldf293f417e1701dd68cc5.png">
</meta>
<meta name="title" content="Viu-TN7-NFT-Minting">
</meta>
<meta name="description" content="Viu-TN7-NFT-Minting">
</meta>
<meta name="author" content="https://viu-tn7-nft-minting.morpheuslabs.io">
</meta>
<meta name="article:publisher" content="
https://viu-tn7-nft-minting.morpheuslabs.io">
</meta>
<meta property="og:locale" content="en_US">
</meta>
<meta property="og:type" content="website">
</meta>
<meta property="og:url" content="https://viu-tn7-nft-minting.morpheuslabs.io">
</meta>
<meta property="og:site_name" content="Viu-TN7-NFT-Minting">
</meta>
<meta property="og:title" content="Viu-TN7-NFT-Minting">
</meta>
<meta property="og:description" content="Viu-TN7-NFT-Minting">
</meta>
<meta property="og:image" content="
https://ml-webstudio-dev.s3.amazonaws.com/solutions/Desktop - 1.png">
</meta>
<meta property="twitter:card" content="summary_large_image">
</meta>
Inspector
Request attributes 2
Request query parameters 0
Request body parameters 0
Request cookies 0
Request headers 9
Response headers 11
Inspector
Notes
Explanations
Custom actions
Search
0 highlights
Search
0 highlights
Done
664,095 bytes | 3,920 millis
Event log (10)
All issues (23)
Memory: 469.0MB

Finding 9: Lack of Rate Limiting and Brute Force Protection

Severity: Medium

Status: Resolved

Description:

The application lacks rate limiting controls on API endpoints, particularly the webhook authentication endpoint. Attackers can send unlimited requests without any restrictions, potentially overwhelming server resources and causing denial of service.

The vulnerability allows attackers to rapidly send multiple requests to the /api/v1/webhooks/gsz5mzmhvlp1zd endpoint without any throttling or blocking mechanisms in place.

Impact:

The absence of rate limiting exposes the application to denial of service attacks. When attackers send excessive requests, they can overwhelm the server's processing capacity, making the service slow or unavailable for legitimate users.

Proof of Concept:

1. Brute force the signature and frequently send invalid signatures.
2. In the end add a valid signature value and observe that the server is not restricting frequent invalid requests.

Remediation:

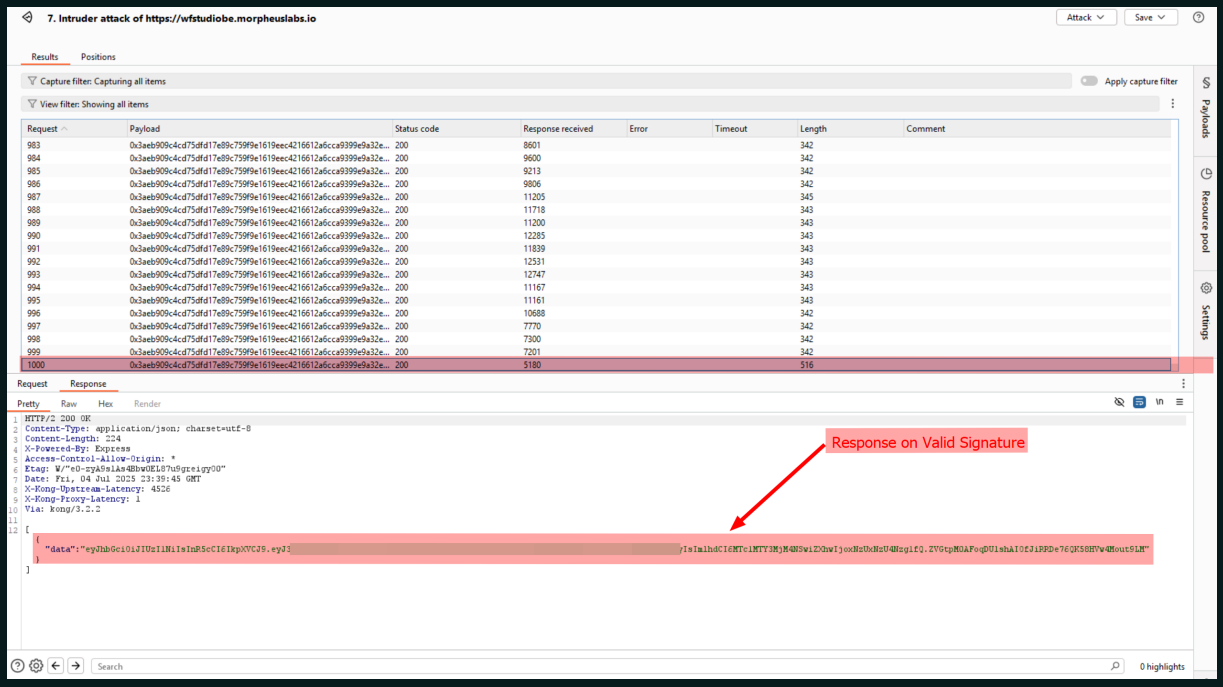
Implement comprehensive rate limiting controls:

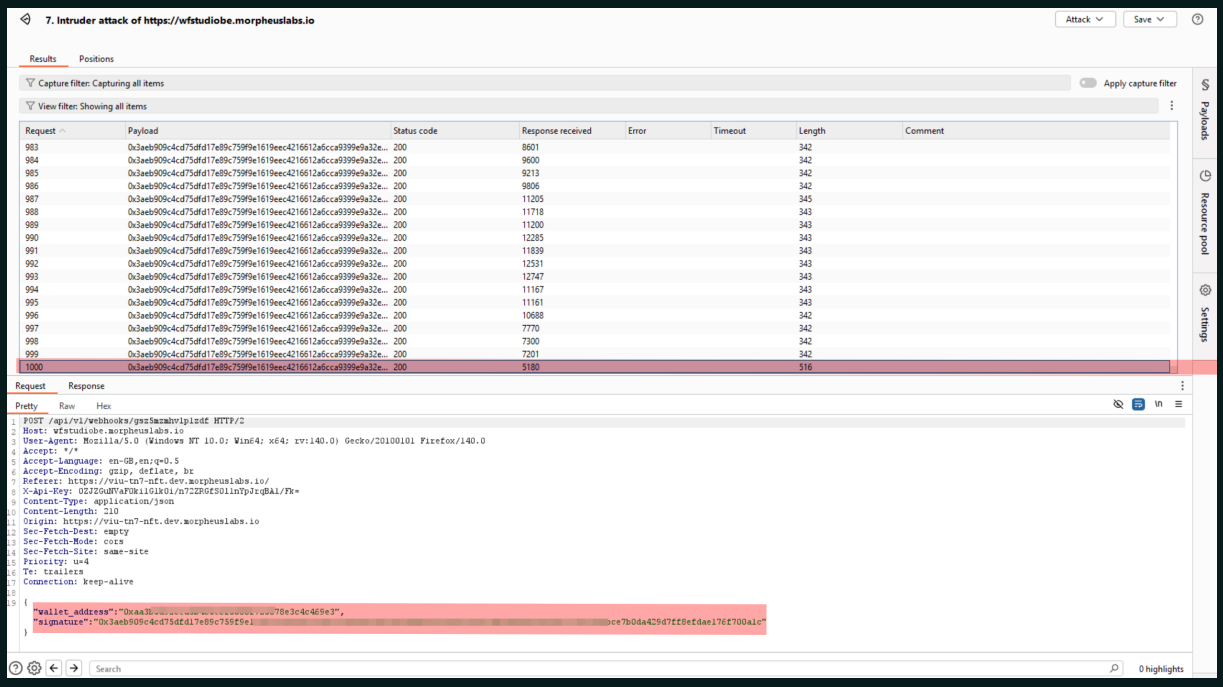
1. API Rate Limiting: Apply rate limiting to all API endpoints, especially those handling authentication and sensitive operations.
2. API Gateway Integration: Deploy API gateways or web application firewalls with built-in rate limiting features for centralized management.
3. Monitoring and Adjustment: Regularly monitor API traffic patterns and adjust rate limits based on legitimate usage patterns.
4. IP Blocking: Automatically block IP addresses that repeatedly exceed rate limits within short time periods.
5. User Feedback: Implement clear error messages and HTTP status codes (e.g., 429 Too Many Requests) to inform users when they exceed limits.

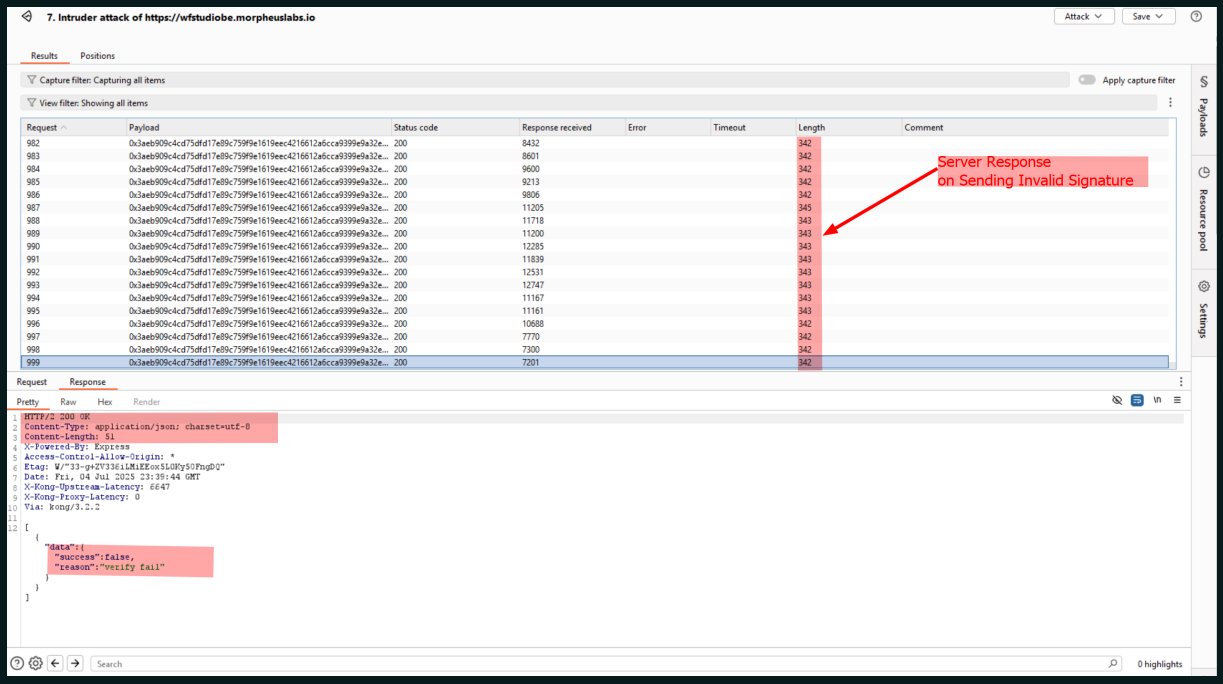
```
Request
POST /api/v1/webhooks/gsz5mzmhvlp1zdf HTTP/2
Host: wfstudiobe.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko
    /20100101 Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://viu-tn7-nft.dev.morpheuslabs.io/
X-Api-Key: 0ZJZGuNVaF0ki1G1k0i/n72ZRGfS0l1nYpJrqBA1/Fk=
Content-Type: application/json
Content-Length: 210
Origin: https://viu-tn7-nft.dev.morpheuslabs.io
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Priority: u=4
Te: trailers
Connection: keep-alive
{"wallet_address":"0xaa3b5d92c469e3","signature":"0x3f700a1c"}
Response
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 224
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"e0-zyA9slAs4Bbw0EL87u9greigyO0"
Date: Fri, 04 Jul 2025 23:39:45 GMT
X-Kong-Upstream-Latency: 4526
X-Kong-Proxy-Latency: 1
Via: kong/3.2.2
[{"data":"eyJhbGciOiJIUzI1yIsImlhdCI6MTc1MTY3MjM4NSwiZXhwIjoxNzUxNzU4Nzg1fQ.
    ZVGtpM0AFoqDUlshAIOfJiRRDe76QK58HVw4Mout9LM"}]
```

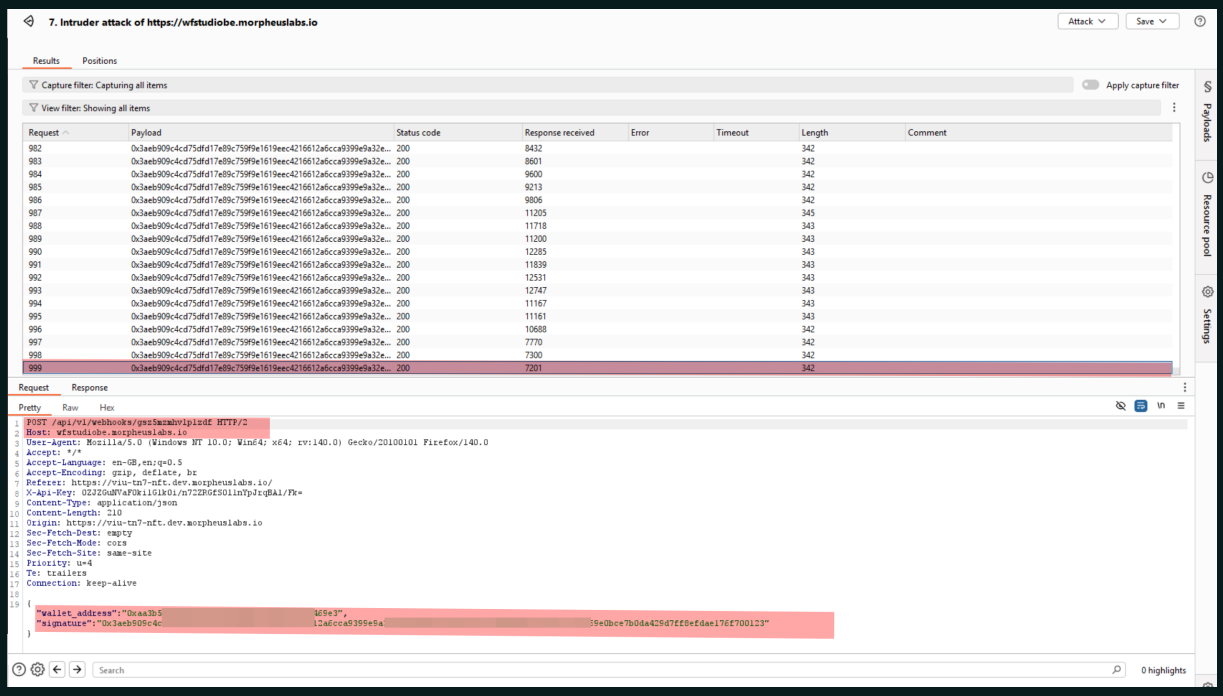
References: - OWASP Identification and Authentication Failures

Attachments:


7. Intruder attack of https://wfstudiobe.morpheuslabs.io
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
Comment
Payloads
Resource pool
Settings
Request
Response
Pretty
Raw
Hex
Render
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 224
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"e0-zyA5slAs4BbwOEL87u9greigyOO"
Date: Fri, 04 Jul 2025 23:39:45 GMT
X-Kong-Upstream-Latency: 4526
X-Kong-Proxy-Latency: 1
Via: kong/3.2.2
"data":"eyJhbGci0iJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
Response on Valid Signature
Search
0 highlights


7. Intruder attack of https://wfstudiobe.morpheuslabs.io
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
Comment
Payloads
Resource pool
Settings
Request
Response
Pretty
Raw
Hex
POST /api/v1/webhooks/gszSmzmhvlplzdf HTTP/2
Host: wfstudiobe.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://viu-tn7-nft.dev.morpheuslabs.io/
X-Api-Key: OZJZGuNVaF0kiiGlk0i/n72ZRGfSOllnYpJrqBAl/Fk=
Content-Type: application/json
Content-Length: 210
Origin: https://viu-tn7-nft.dev.morpheuslabs.io
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Priority: u=4
Te: trailers
Connection: keep-alive
"wallet_address":"0xaa3
"signature":"0x3aeb909c4cd75dfd17e89c759f9e1
Search
0 highlights


7. Intruder attack of https://wfstudiobe.morpheuslabs.io
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
Comment
Server Response
on Sending Invalid Signature
Payloads
Resource pool
Settings
Request
Response
Pretty
Raw
Hex
Render
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 51
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"33-g+ZV33iLMiEXox5L0Ky50FngDQ"
Date: Fri, 04 Jul 2025 23:39:44 GMT
X-Kong-Upstream-Latency: 6647
X-Kong-Proxy-Latency: 0
Via: kong/3.2.2
[
{
"data":{
"success":false,
"reason":"verify fail"
}
}
]
Search
0 highlights


7. Intruder attack of https://wfstudiobe.morpheuslabs.io
Attack
Save
Results
Positions
Capture filter: Capturing all items
Apply capture filter
View filter: Showing all items
Request
Payload
Status code
Response received
Error
Timeout
Length
Comment
Payloads
Resource pool
Settings
Request
Response
Pretty
Raw
Hex
POST /api/v1/webhooks/gsz5mzmhvlp1zdf HTTP/2
Host: wfstudiobe.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://viu-tn7-nft.dev.morpheuslabs.io/
X-Api-Key: OZJZGuNVaFUki1Glk0i/n72ZRGfSOlinYpJrqBAl/Fk=
Content-Type: application/json
Content-Length: 210
Origin: https://viu-tn7-nft.dev.morpheuslabs.io
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Priority: u=4
Te: trailers
Connection: keep-alive
{
"wallet_address":"0xaa3b5 4E5e3",
"signature":"0x3aeb9O9c4c 12a6cca9399e9a 39e0bce7b0da425d7ff8efdae176f700123"
}
Search
0 highlights

Finding 10: Always Returning HTTP 200 on Errors

Severity: Low

Status: Acknowledged

Description:

Webhook responds 200 OK even when backend detects validation errors. Uniform 200 responses mask failures, preventing front-end or monitoring systems from detecting issues and enabling automated error exploitation.

Negative tests (invalid integer, amount 0, etc.) still delivered HTTP 200 with error objects nested in JSON.

Impact:

1. Front-end may treat failed payments as success, leading to broken UX and inconsistent state.
2. Security scanners may miss real issues due to misleading status codes.

Proof of Concept:

```
curl -X POST … -d '{"amount":"0"}' # → HTTP 200, body: { "error": "Invalid integer" }
```

Remediation:

Implement proper HTTP status codes:

1. Return proper HTTP semantics (400, 401, 422, etc.).
2. Ensure client-side code checks both status code and body.
3. Implement consistent error handling across all endpoints.
4. Add proper error logging and monitoring.

Finding 11: Hard Coded API Key Disclosure

Severity: Low

Status: Resolved

Description:

The application exposes API keys directly in client-side JavaScript code. The `APP_X_API_KEY` constant is hardcoded in the frontend code, making it visible to anyone who inspects the page source or network traffic.

Impact:

Exposed API keys can lead to unauthorized service usage, increased billing charges, and potential quota exhaustion that affects legitimate users. Attackers could exploit the key to access third-party services or manipulate application functionality.

Proof of Concept:

Browse the following URL and Search for "APP_X_API_KEY" keyword in response: https://viu-tn7-nft.dev.morpheuslabs.io

Remediation:

Implement secure API key management:

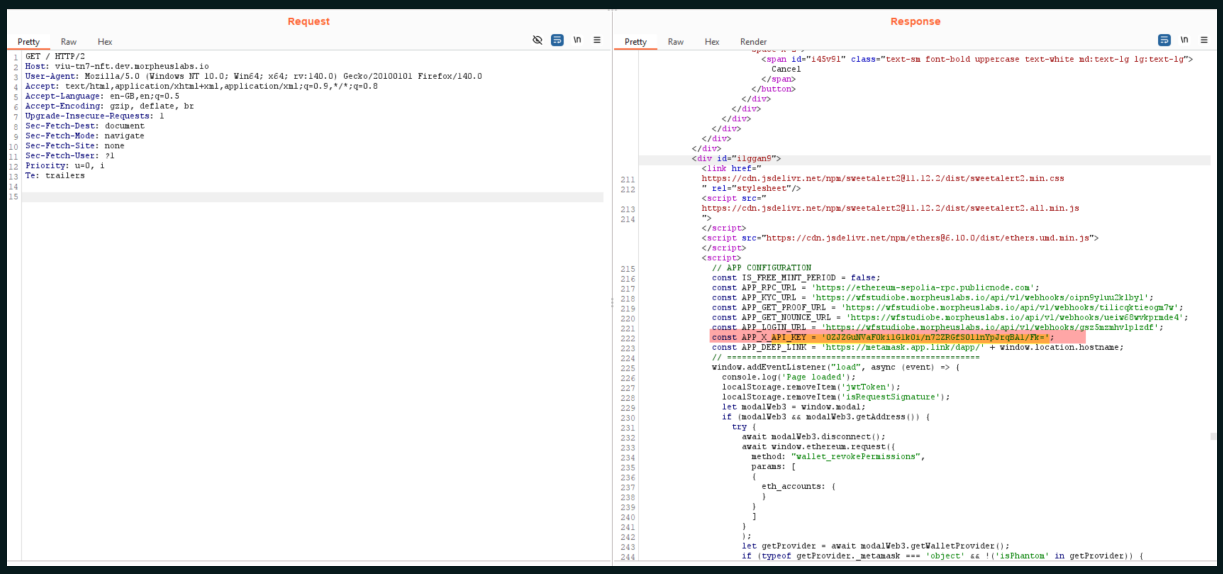
1. Server-Side Storage: Move API keys to server-side storage and proxy requests through secure endpoints instead of exposing them in client-side code.
2. Key Restrictions: Limit API key permissions to only necessary services and restrict access to specific IP addresses or referrers.
3. Key Rotation: Immediately rotate any exposed API keys and implement regular rotation schedules.
4. Monitoring: Set up billing alerts and usage quotas to detect unusual activity quickly.
5. Environment Variables: Use environment variables or secure storage solutions for managing API keys during development.

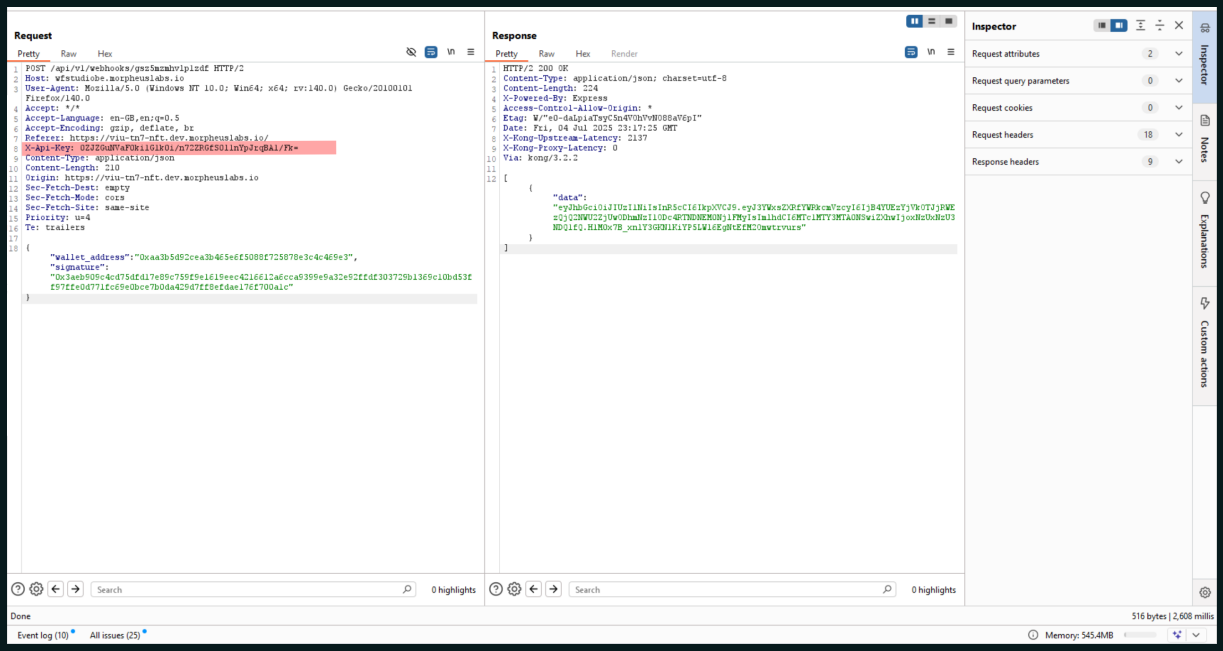
```
Request
GET / HTTP/2
Host: viu-tn7-nft.dev.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko
    /20100101 Firefox/140.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Upgrade-Insecure-Requests: 1
```

```
 9      Sec-Fetch-Dest: document
10      Sec-Fetch-Mode: navigate
11      Sec-Fetch-Site: none
12      Sec-Fetch-User: ?1
13      Priority: u=0, i
14      Te: trailers
15      Response
16      HTTP/2 200 OK
17      Content-Type: text/html; charset=UTF-8
18      Content-Length: 724241
19      X-Amz-Id-2:
20      KWYWrjlA3gzDbziZ8/X8ULbHamVvacDhtAEnz6o8AHI8albkikboEloOYNMhuxtLx1C4j6T
21        0Xfk2L3LvJpQmr04y8PDwSAwS2Phh1L8bS7k=
22      X-Amz-Request-Id: 60Z7Z1FZ44JJS1GA
23      Date: Thu, 03 Jul 2025 15:50:51 GMT
24      Last-Modified: Thu, 26 Jun 2025 02:15:35 GMT
25      Etag: "4f64b7d1eeb3b08234800fadeea8d9b6"
26      Server: AmazonS3
27      X-Kong-Upstream-Latency: 29
28      X-Kong-Proxy-Latency: 0
29      Via: kong/3.2.2
30      <!doctype html> <html lang="en"><head><meta charset="UTF-8"> ...
31      const APP_X_API_KEY = '0ZJZGuNVaF0ki1G1k0i/n72ZRGfSOl1nYpJrqBA1/Fk=';
32      ...
33      </script></html>
```

References: - OWASP Identification and Authentication Failures

Attachments:




Request
Pretty Raw Hex
POST /api/v1/webhooks/gsz5mzmhvlplzdf HTTP/2
Host: wfstudiobe.morpheuslabs.io
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:140.0) Gecko/20100101 Firefox/140.0
Accept: */*
Accept-Language: en-GB,en;q=0.5
Accept-Encoding: gzip, deflate, br
Referer: https://viu-tn7-nft.dev.morpheuslabs.io/
X-Api-Key: OZJZ0uNVaFUkiiGlkOi/n72ZRGfS01lnYpJrqBAl/Fk=
Content-Type: application/json
Content-Length: 210
Origin: https://viu-tn7-nft.dev.morpheuslabs.io
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-site
Priority: u=4
Te: trailers
{
"wallet_address":"0xaa3b5d92cea3b465e6f5088f725878e3c4c469e3",
"signature":
"0x3aeb509c4cd75dfd17e89c759f9e1619eec421661 2a6cca9399e9a32e92ffdf30372 9b1369c10bd53f f97ffe0d771fc69e0bce7b0da429d7ff8efdae176f700a1c"
}
Response
Pretty Raw Hex Render
HTTP/2 200 OK
Content-Type: application/json; charset=utf-8
Content-Length: 224
X-Powered-By: Express
Access-Control-Allow-Origin: *
Etag: W/"e0-daLpiaTsyC5n4V0hVvNO88aV6pI"
Date: Fri, 04 Jul 2025 23:17:25 GMT
X-Kong-Upstream-Latency: 2137
X-Kong-Proxy-Latency: 0
Via: kong/3.2.2
[
{
"data":
"eyJhbGci0iJIUzIiNiIsInR5cCI6IkpXVCJ9.eyJ3YWxsZXRfYWRkcmVzcyI6IjB4YUEzYjVkOTJjRWEzQjQ2NWU2ZjUwODhmNzI1ODc4RTNDNEM0NjlFMyIsImlhdCI6MTc1MTY3MTA0NSwiZXhwIjoxNzUxNzU3ND QlfQ.HiMOx7B_xnlY3GKNlKiYP5LW16EgNrEfM20mwtrvurs"
}
]
Inspector
Request attributes 2
Request query parameters 0
Request cookies 0
Request headers 18
Response headers 9
Inspector
Notes
Explanations
Custom actions
Search
0 highlights
Search
0 highlights
Done
516 bytes | 2,608 millis
Event log (10)
All issues (25)
Memory: 545.4MB

Disclaimer

This security report (“Report”) is provided by FailSafe (“Security Provider”) for the exclusive use of the client (“Client”). The security scope is limited to a technical review of the application code supplied by the Client.

While FailSafe has made every effort to identify vulnerabilities and deviations from best practices, we do not guarantee the absence of all security issues or that the application will function as intended in every environment.

FailSafe is not liable for any losses or damages arising from the use, misuse, or inability to use the application. This Report is not an endorsement or certification of the application and may not be shared or reproduced without FailSafe’s written consent.

The Client is solely responsible for the deployment, operation, and further testing of the application, as well as for implementing any recommended changes. FailSafe may update this Report if new information becomes available, and the Client is encouraged to maintain ongoing security reviews.

By using this Report, the Client accepts these terms and conditions.