

CyberHAT Security Team

# External penetration test report for {REDACTED} project



---

# Contents

<b>Contents</b> .....	<b>2</b>
<b>Executive summary</b> .....	<b>3</b>
<b>Testing details</b> .....	<b>3</b>
Reconnaissance.....	3
Discovery, Perimeter, Stateful Firewall and DNS Analysis.....	3
Result summary.....	4
Vulnerability Scanning.....	4
Scanners used.....	4
Summary of Scanning results.....	5
<b>Penetration testing</b> .....	<b>5</b>
Objectives.....	5
Network and Host Test Coverage: Common Network and Host Configuration Issues.....	5
Network and Host Test Coverage: Encryption.....	5
Application Test Coverage: Information Disclosure.....	6
Application Test Coverage: Authentication.....	6
Application Test Coverage: Authorization.....	7
Application Test Coverage: Data Validation - Reflection Issues.....	7
Application Test Coverage: Data Validation – Injection and Miscellaneous.....	7
Application Test Coverage: Session Handling.....	8
Application Test Coverage: Application Server Configuration Issues.....	8
Wireless Network Test Coverage.....	9
Social Engineering Test Coverage.....	9
<b>Findings Details</b> .....	<b>9</b>
<b>Summary</b> .....	<b>13</b>

## Executive summary

The first objective of this penetration test was to fully examine {REDACTED COMPANY} systems and services to identify vulnerabilities that could allow an attacker to compromise the confidentiality, integrity or availability of those systems and services.

Our second objective was to prove exploitability of vulnerabilities

Third objective was to give recommendations to remediate detected issues.

---

**Note:** Due to priority of objectives not all of the issues were tested in full range of their potential impact. Full exploitation was not pursued if the vulnerability appeared to be systemic or if remediation was mandatory for PCI compliance, or if exploitation would have jeopardized either full test coverage or the stability of the systems under test.

## Testing details

### Reconnaissance

Reconnaissance step was conducted encompassing both active and passive techniques using Whois queries, Search engines, amass, DNSDumpster, gobuster and other web services and tools.

### Discovery, Perimeter, Stateful Firewall and DNS Analysis

At a minimum, an analysis was conducted from an external host to the target network with use of VPN profile allowing access inside of the target network. In this section we give a short summary of results, full results will be included in next sections.

Next targets were in scope in two isolated environments:

- {REDACTED HOST}
- {REDACTED HOST}
- {REDACTED HOST}
- {REDACTED HOST}
- {REDACTED HOST}

Also, separate part of penetration testing was a white-box examining of smart-contracts with use of source code from repository

### Result summary

Based upon stateful firewall inspection tests, DNS queries, port scans and services identified (also tested for common misconfigurations or vulnerabilities), the network devices are well secured.

---

## Vulnerability Scanning

### Scanners used

Tool	Version	Wordlist/modules/comments
ZAP	2.12.0	Active and passive scans
Nikto	2.1.6	
Nmap	7.94	default, vuln, intrusive, brute, discovery and auth scripts
nuclei	2.9.6	
Burp Professional	2023.4.5	Intruder, Repeater, Scanner, Authorize, Additional Checks for Scanner, wfuzz wordlists
Gobuster	3.5.0	dirb big wordlist, subdomains wordlist (github danTaler/Wordlists)
sqlmap	1.6	risk 3 and level 5
dotdotpwn	3.0.2	
xsser	1.8.4	
slither	v0.9.5	
Manticore	0.3.7	

### Summary of Scanning results

Lot of low-severity findings such as missing Security Headers were detected, but also our team detected a small amount of more severe vulnerabilities such as seed regeneration and 2FA bruteforcing

## Penetration testing

---

---

## Objectives

The first objective was maximum test coverage; the second objective was safeguarding the stability of the systems under test, and the last objective was proof of exploitability. The priority of these objectives dictated that vulnerabilities were not necessarily pursued to the point of full exploitation and compromise. Full exploitation was not pursued if the vulnerability appeared to be systemic, or if remediation was mandatory by reason of compliance drivers, or if exploitation would have jeopardized either full test coverage or the stability of the systems under test.

## Network and Host Test Coverage: Common Network and Host Configuration Issues

Deprecated or vulnerable services	No faults found.
Open Administrative interfaces	No faults found.
Services open outside the VPN	No faults found.
Authentication Attacks	No faults found.

## Network and Host Test Coverage: Encryption

Transport Protocol	No faults found.
Transport Cipher Suites Support	No faults detected.
Clear Text Transport of Sensitive Data	Potentially possible, but with very small chance (Strict Transport Security Headers not set in some places)

## Application Test Coverage: Information Disclosure

Robots.txt	No faults detected.
------------	---------------------

---

Comments	No faults detected.
Hidden Files	No faults detected.
Error Handling	No faults detected.

### **Application Test Coverage: Authentication**

User Account Enumeration	No faults found.
Guessable Accounts	No faults found.
Brute Force and Account Lockout	2FA bruteforcing
Authentication Bypass	No faults found.
Password Recovery and Reset	No faults found.
Password Complexity	No faults found.
Secure Logout	No faults found.
Browser Caching	No faults found.
CAPTCHA Devices	No faults found.
MFA	2FA bypassing
Race Conditions	No faults found.

### **Application Test Coverage: Authorization**

Path Traversal	No faults found.
Authorization Bypass	No faults found.
Privilege Escalation	No faults found.
RBAC	No faults found.

---

## Application Test Coverage: Data Validation - Reflection Issues

Reflected XSS	No faults found.
Persistent XSS	No faults found.
DOM Based XSS	No faults found.
XSS Flashing	No faults found.
Input Reflected in output	No faults found.

## Application Test Coverage: Data Validation – Injection and Miscellaneous

SQL Injection	No faults found.
LDAP Injection	No faults found.
ORM Injection	No faults found.
XML Injection	No faults found.
SSI Injection	No faults found.
XPath Injection	No faults found.
IMAP/SMTP Injection	No faults found.
Code Injection	No faults found.
OS Commanding	No faults found.
Buffer overflow	No faults found.
File upload vulnerabilities	No faults found.
HTTP Splitting/Smuggling	No faults found.
DoS	No faults found.
Smart contract security	No faults found.

---

## Application Test Coverage: Session Handling

Session Predictability	No faults found.
Encrypted transport	Minor findings
Cookie Attributes	No faults found.
Session Fixation	No faults found.
Session Re-Use	No faults found.
Cache Control	No faults found.
CSRF Vulnerabilities	No faults found.

## Application Test Coverage: Application Server Configuration Issues

File Extensions Handling	No faults found.
Old, Backup and Unreferenced Files	No faults found.
HTTP Methods and XST	No faults found.

## Wireless Network Test Coverage

Out of scope.

## Social Engineering Test Coverage

Out of scope.



---

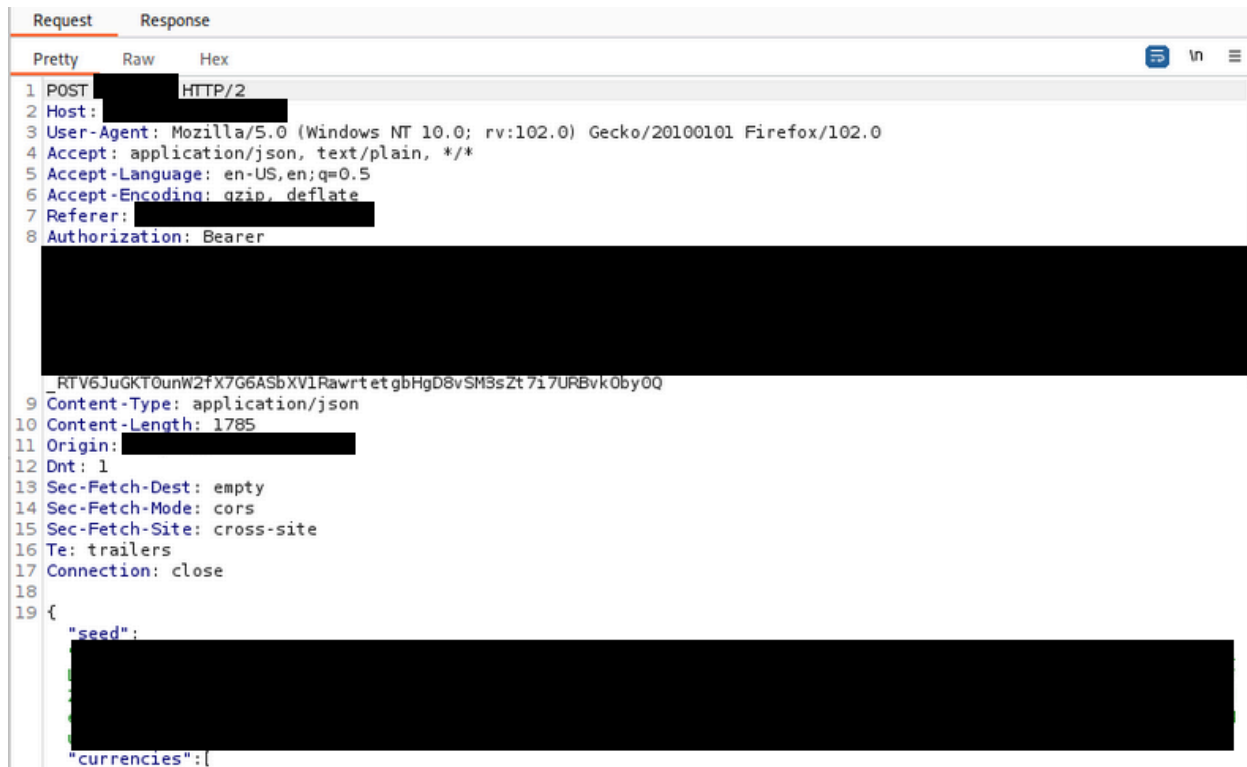
## Findings Details

**Finding:** seed phrase regeneration

**Severity:** High

**Target:** {REDACTED TARGET}

**Description:** Seed is a very important sensitive information which must be properly protected. The problem is possibility to change seed any moment. The request to set new seed is:



```
Request  Response
Pretty  Raw    Hex
1 POST [REDACTED] HTTP/2
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: [REDACTED]
8 Authorization: Bearer [REDACTED]
[REDACTED]
RTV6JuGKT0unW2fX7G6ASbXV1RawrtetgbHgD8vSM3sZt7i7URBvk0by0Q
9 Content-Type: application/json
10 Content-Length: 1785
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: cross-site
16 Te: trailers
17 Connection: close
18
19 {
  "seed": [REDACTED]
  "currencies": [
```

And response status code is 201 Created:

```
Request  Response
Pretty  Raw    Hex    Render
1 HTTP/2 201 Created
2 Date: Mon, 03 Jul 2023 17:04:07 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 787
5 X-Powered-By: Express
6 Access-Control-Allow-Origin: [REDACTED]
7 Vary: Origin
8 Etag: W/"313-c573gVUUCVHp83GRUYEV3+xxfRU"
9 Cf-Cache-Status: DYNAMIC
10 Report-To: [REDACTED]
11 Nel: [REDACTED]
12 Server: cloudflare
13 Cf-Ray: 7e10b4ab8d983678-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
  "token": [REDACTED]
  "is2FASet":true,
  "isSeedSet":true
}
```

As we can see, it's possible to send new request and regenerate seed for user with certain access token.

For example, it's impossible in other modules, such as exchanger or multisender:

```
Request  Response
Pretty  Raw    Hex
1 POST /auth/backup HTTP/2
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: [REDACTED]
8 Authorization: Bearer [REDACTED]
9 Content-Type: application/json
10 Content-Length: 1327
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: cross-site
16 Te: trailers
```



---

**Finding:** Insecure 2FA implementation

**Severity:** Medium

**Description:** The OTP code of 2FA can be bruteforce because there are no rate limits. There is no any protection against otp bruteforce, so attacker's success on getting OTP sequence depends only on spent time and attacker's luck.

206	000205	400	<input type="checkbox"/>	<input type="checkbox"/>	769
207	000206	400	<input type="checkbox"/>	<input type="checkbox"/>	771
208	000207	400	<input type="checkbox"/>	<input type="checkbox"/>	773
209	000208	400	<input type="checkbox"/>	<input type="checkbox"/>	763
210	000209	400	<input type="checkbox"/>	<input type="checkbox"/>	763
211	000210	400	<input type="checkbox"/>	<input type="checkbox"/>	769
212	000211	400	<input type="checkbox"/>	<input type="checkbox"/>	769
213	000212	400	<input type="checkbox"/>	<input type="checkbox"/>	773
214	000213	400	<input type="checkbox"/>	<input type="checkbox"/>	769
215	000214	400	<input type="checkbox"/>	<input type="checkbox"/>	761
216	000215	400	<input type="checkbox"/>	<input type="checkbox"/>	765
217	000216	400	<input type="checkbox"/>	<input type="checkbox"/>	765
218	000217	400	<input type="checkbox"/>	<input type="checkbox"/>	773

```
1 HTTP/2 400 Bad Request
2 Date: Tue, 04 Jul 2023 19:34:51 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 67
5 X-Powered-By: Express
6 Access-Control-Allow-Origin: [REDACTED]
7 Vary: Origin
8 Etag: W/"43-omYpWgbYmALp/iCM067pqkzh6vU"
9 Cf-Cache-Status: DYNAMIC
10 Report-To: [REDACTED]
11 Nel: [REDACTED]
12 Server: cloudflare
13 Cf-Ray: 7e19ced8dad9202-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
17   "statusCode": 400,
18   "message": "Wrong 2fa code",
19   "error": "Bad Request"
20 }
```

**Remediation:** We need to set limit on attempts to validate request. For example, it may be 5 attempts per minute.

---

**Finding:** Strict transport security not enforced

**Severity:** Low

**Targets:** {REDACTED TARGETS}

**Description:** The application fails to prevent users from connecting to it over unencrypted connections. An attacker able to modify a legitimate user's network traffic could bypass the application's use of SSL/TLS encryption, and use the application as a platform for attacks against its users. This attack is performed by rewriting HTTPS links as HTTP, so that if a targeted user follows a link to the site from an HTTP page, their browser never attempts to use an encrypted connection. The sslstrip tool automates this process.

**Finding:** Cross-site request forgery

**Severity:** Medium

**Target:** {REDACTED TARGET}

**Description:** Cross-site request forgery (CSRF) vulnerabilities may arise when applications rely solely on HTTP cookies to identify the user that has issued a particular request. Because browsers automatically add cookies to requests regardless of their origin, it may be possible for an attacker to create a malicious web site that forges a cross-domain request to the vulnerable application.

**Remediation:** validate that Host and Referer headers in relevant requests are both present and contain the same domain name

```
Advisory Request1 Response1 Request2 Response2
Pretty Raw Hex
1 POST [REDACTED] HTTP/2
2 Host: [REDACTED]
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; rv:102.0) Gecko/20100101 Firefox/102.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Referer: [REDACTED]
8 Authorization: Bearer [REDACTED]
9 Content-Type: multipart/form-data;
boundary=-----26263403201250535077820538259
10 Content-Length: 473
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
? ⚙️ ⏪ ⏩ Search... 1 highlight
```

```
Advisory Request1 Response1 Request2 Response2
Pretty Raw Hex Render
1 HTTP/2 201 Created
2 Date: Tue, 04 Jul 2023 15:34:08 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 266
5 X-Powered-By: Express
6 Access-Control-Allow-Origin: [REDACTED]
7 Vary: Origin
8 Etag: W/"10a-YHepuIGQlM2N4Ent97ch6GdUKn8"
9 Cf-Cache-Status: DYNAMIC
10 Report-To: [REDACTED]
11 Nel: [REDACTED]
12 Server: cloudflare
13 Cf-Ray: 7e186e36ea72367b-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
  "duplicatesCounters": {
    [REDACTED]
  }
}
```

---

## Summary

{REDACTED CLIENT} is a well-protected service with only one high severity issue which is seed regeneration. Next, medium severity issue - is a 2FA bruteforcing, which gives attacker possibility to bypass 2FA protection. And minor low severity finding is unenforced strict transport security, which could be used in MITM attacks.