



# WEB APPLICATION PENETRATION TESTING REPORT

for

## IT Soft Solutions NV



Compliance with  
UnderDefense Certification criteria:  
**Meets criteria**

August 2025

# Table Of Contents

|                                                               |           |
|---------------------------------------------------------------|-----------|
| <b>Table Of Contents</b>                                      | <b>1</b>  |
| <b>Executive Summary</b>                                      | <b>2</b>  |
| 1.1 Project Objectives                                        | 3         |
| 1.2 Scope & Timeframe                                         | 3         |
| 1.2.1 Hostname                                                | 3         |
| 1.2.2 User Accounts provided by IT Soft Solutions NV          | 4         |
| 1.3 Summary of Findings                                       | 5         |
| 1.4 Summary of Business Risks                                 | 6         |
| 1.5 High-Level Recommendations                                | 7         |
| <b>Technical Details</b>                                      | <b>8</b>  |
| 2.1 Methodology                                               | 8         |
| 2.2 Security tools used                                       | 8         |
| 2.3 Project limitations                                       | 8         |
| <b>Findings Details</b>                                       | <b>9</b>  |
| 3.1 Low severity findings                                     | 9         |
| 3.1.1 User enumeration                                        | 9         |
| 3.1.1.1 Reset Password                                        | 9         |
| 3.1.1.2 Signup                                                | 11        |
| 3.1.2 Improper Captcha Implementation                         | 13        |
| 3.2 Informational severity findings                           | 15        |
| 3.2.1 CORS misconfiguration                                   | 15        |
| 3.2.2 Weak password policies                                  | 17        |
| <b>APPENDIX A - Performed tests according to OWASP Top 10</b> | <b>18</b> |

## Executive Summary

This report presents the results of the “Black Box” penetration testing for the IT Soft Solutions NV Web application. The recommendations provided in this report are structured to facilitate remediation of the identified security risks. This document serves as a formal letter of attestation for the recent IT Soft Solutions NV “Black Box” Web penetration testing.

Evaluation ratings compare information gathered during the engagement to “best in class” criteria for security standards. We believe that the statements made in this document provide an accurate assessment of the IT Soft Solutions NV current security.

We highly recommend reviewing the Summary section of business risks and High-Level Recommendations for a better understanding of risks and discovered security issues.

|                     |                 |
|---------------------|-----------------|
| Scope of assessment | Web Application |
| Security Level      | <b>A</b>        |
| Grade               | Excellent       |

| Grade    | Security     | Criteria Description                                                                                                                                                                                               |
|----------|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>A</b> | Excellent    | The security exceeds “Industry Best Practice” standards. The overall posture was found to be excellent with only a few low-risk findings identified.                                                               |
| <b>B</b> | Good         | The security meets accepted standards for “Industry Best Practice.” The overall posture was found to be strong with only a handful of medium- and low-risk shortcomings identified.                                |
| <b>C</b> | Fair         | Current solutions protect some areas of the enterprise from security issues. Moderate changes are required to elevate the discussed areas to “Industry Best Practice” standards                                    |
| <b>D</b> | Poor         | Significant security deficiencies exist. Immediate attention should be given to the discussed issues to address exposures identified. Major changes are required to elevate to “Industry Best Practice” standards. |
| <b>F</b> | Unacceptable | Serious security deficiencies exist. Shortcomings were identified throughout most or even all of the security controls examined. Improving security will require a major allocation of resources.                  |

## 1.1 Project Objectives

Our primary goal within this project was to provide the IT Soft Solutions NV with an understanding of the current level of security in the web application and its infrastructure components. We completed the following objectives to accomplish this goal:

- Identifying application-based threats to and vulnerabilities in the application
- Comparing IT Soft Solutions NV current security measures with industry best practices
- Providing recommendations that IT Soft Solutions NV can implement to mitigate threats and vulnerabilities and meet industry best practices

The Common Vulnerability Scoring System (CVSS) version 4.0 was used to calculate the scores of the vulnerabilities found. When calculating the score, the following CIA provision, supplied by the IT Soft Solutions NV has been taken into account:

| Scope             | Confidentiality | Integrity | Availability |
|-------------------|-----------------|-----------|--------------|
| All scope objects | High            | High      | High         |

## 1.2 Scope & Timeframe

Testing and verification were performed between July 2, 2025 and July 16, 2025. The scope of this project was limited to the IT Soft Solutions NV Web application and the specific infrastructure on which the application resides. We conducted the tests using a non-production version of the IT Soft Solutions NV Web application. All other applications and servers were out of scope. All testing and verification were conducted from outside of IT Soft Solutions NV offices.

The following hosts were considered to be in scope for testing.

### 1.2.1 Hostname

| Scope:                                                                        | Description:     |
|-------------------------------------------------------------------------------|------------------|
| <a href="https://www.demobroadway.com/en">https://www.demobroadway.com/en</a> | Demo Environment |

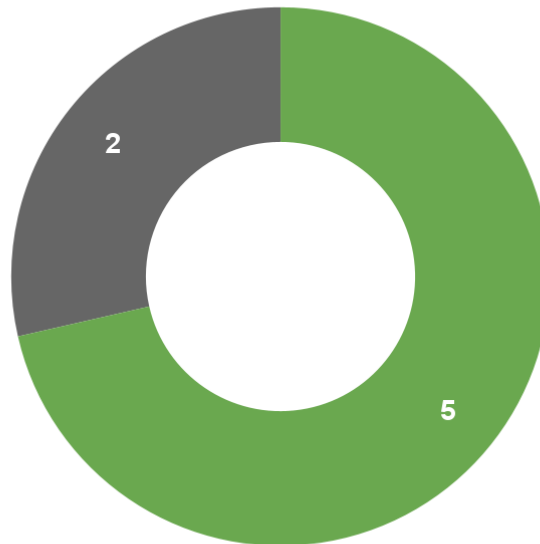
### 1.2.2 User Accounts provided by IT Soft Solutions NV

| Asset:           | Username                                  |
|------------------|-------------------------------------------|
| Demo environment | ilya.z@underdefense.com                   |
| Demo environment | ilya.z+2@underdefense.com                 |
| Demo environment | ilya.z+3@underdefense.com                 |
| Demo environment | marian.sadovskyi+player1@underdefense.com |
| Demo environment | marian.sadovskyi+player2@underdefense.com |

## 1.3 Summary of Findings

Our assessment of the IT Soft Solutions NV application revealed the following vulnerabilities:

Vulnerabilities by locations



● Critical ● High ● Medium ● Low ● Informational

Security experts performed manual security testing according to the OWASP Web Application Testing Methodology, which demonstrates the following results.

| Severity                      | Critical | High | Medium | Low | Informational |
|-------------------------------|----------|------|--------|-----|---------------|
| Number of issues by types     | 0        | 0    | 0      | 2   | 2             |
| Number of issues by locations | 0        | 0    | 0      | 5   | 2             |

Severity scoring:

- **Critical** – Immediate threat to key business processes.
- **High** – Direct threat to key business processes.
- **Medium** – Indirect threat to key business processes or partial threat to business processes.
- **Low** – No direct threat exists. The vulnerability may be exploited using other vulnerabilities.
- **Informational** – This finding does not indicate vulnerability, but states a comment that notifies about design flaws and improper implementation that might cause a problem in the long run.

## 1.4 Summary of Business Risks

In the case of IT Soft Solutions NV web application

**Low** and **Informational** severity issues can lead to:

- Increased likelihood of credential stuffing and phishing attacks due to user enumeration could affect user trust and brand reputation.
- Elevated risk of unauthorized access through brute-force attacks on known user accounts.
- Greater exposure to automated abuse from bots due to improper CAPTCHA validation.
- Potential for mass account creation and exploitation of password reset functionality.
- Risk of compromised user data confidentiality from cross-origin API access due to CORS misconfiguration.
- Could be chained with other vulnerabilities to increase potential impact.

## 1.5 High-Level Recommendations

Taking into consideration all issues that have been discovered, we highly recommend to:

- Implement proper server-side CAPTCHA validation on all authentication-related endpoints.
- Standardize and strengthen password policy to align with industry best practices.
- Allow only designated origins and resources to interact with your application to prevent unauthorized cross-domain access.
- Return the same generic messages regardless if the username exists or not.

# Technical Details

## 2.1 Methodology

Our Penetration Testing Methodology grounded on the following guides and standards:

- [Penetration Testing Execution Standard \(PTES\)](#)
- [OWASP Top 10 Application Security Risks](#)
- [OWASP Web Security Testing Guide](#)
- [Open Source Security Testing Methodology Manual \(OSSTMM\)](#)

Penetration Testing Execution Standard (PTES) consists of seven main sections which start from the initial communication and reasoning behind a pentest, through intelligence gathering and threat modeling phases where testers are working behind the scenes to get a better understanding of the tested organization, through vulnerability research, exploitation and post-exploitation, where the technical security expertise of the testers come to play and combine with the business understanding of the engagement, and finally to the reporting, which captures the entire process.

Open Web Application Security Project (OWASP) is an industry initiative for web application security. OWASP has identified the 10 most common attacks that succeed against web applications. Besides, OWASP has created Application Security Verification Standard (ASVS) which helps to identify threats, provides a basis for testing web application technical security controls, and can be used to establish a level of confidence in the security of Web applications.

The Open Source Security Testing Methodology Manual (OSSTMM) is peer-reviewed and maintained by the Institute for Security and Open Methodologies (ISECOM). It has been primarily developed as a security auditing methodology assessing against regulatory and industry requirements. It is not meant to be used as a standalone methodology but rather to serve as a basis for developing one which is tailored toward the required regulations and frameworks.

## 2.2 Security tools used

- *Manual testing:* Burp Suite Pro
- *Vulnerability scan:* Nessus, nikto
- *Directory enumeration:* gobuster, dirsearch, ffuf
- *Injection testing tools:* XSSHunter, SQLmap

## 2.3 Project limitations

The assessment was conducted against a testing environment with all limitations it provides and valid on the date of the report submission hereto. The description of findings, recommendations, and risks was valid on the date of submission the report hereto. Any projection to the future of the report's information is subject to risk due to changes in the application architecture, and it may no longer reflect its logic and controls.

# Findings Details

## 3.1 Low severity findings

### 3.1.1 User enumeration

Severity: **Low**

Location:

- <https://api.demobroadway.com/api/v1/public/security/auth/signup>
- <https://api.demobroadway.com/api/v1/public/player/send-reset-link>

Impact:

This vulnerability allows an attacker to determine whether a specific email address is registered in the system. By automating signup and reset-password attempts and analyzing the server responses, an attacker can enumerate valid user accounts. This significantly increases the risk of targeted attacks such as credential stuffing, phishing, and brute-force attacks against known users.

Vulnerability Details:

User enumeration occurs when an application reveals whether a user exists based on differences in its responses. It is commonly found in login, registration, and password reset functionalities. In this case, the application exposes this vulnerability by returning distinct responses for existing and non-existing email addresses during signup and password reset processes.

Additionally, the lack of CAPTCHA validation on these endpoints allows an attacker to automate requests without restriction, making enumeration attacks easier and more scalable.

Proof of Vulnerability:

#### 3.1.1.1 Reset Password

The application sends different responses depending on whether the user exists.

1. A password reset request for an invalid user.

#### HTTP request:

```
POST /api/v1/public/player/send-reset-link HTTP/2
Host: api.demobroadway.com
Content-Length: 2474
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://www.demobroadway.com
Referer: https://www.demobroadway.com/
Accept-Encoding: gzip, deflate, br
```

Connection: keep-alive

```
{"email": "marian.sadovskyi+pla499@underdefense.com", "captchaToken": "[REDACTED]"}
```

#### HTTP response:

```
HTTP/2 200 OK
Content-Type: application/json
Date: Fri, 11 Jul 2025 09:03:31 GMT
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
X-Cache: Miss from cloudfront
Via: 1.1 043fc2faaa02eeb59193e3fa300adb6a.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: AMS1-C1
Alt-Svc: h3=":443"; ma=86400
X-Amz-Cf-Id: PD7zWTdbzLrmFrIy4ZJ4iLmrgKL41q1ZCmp0xWsV3BzBCBM1PnGFPA==
```

```
{"code":401,"message":"Player with email='marian.sadovskyi+pla499@underdefense.com' doesn't exist."}
```

## 2. A password reset request for an existing user.

#### HTTP request:

```
POST /api/v1/public/player/send-reset-link HTTP/2
Host: api.demobroadway.com
Content-Length: 2475
User-Language: en
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://www.demobroadway.com
Referer: https://www.demobroadway.com/
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

```
{"email": "marian.sadovskyi+player1@underdefense.com", "captchaToken": "[REDACTED]"}
```

#### HTTP response:

```
HTTP/2 200 OK
Content-Type: application/json
Date: Fri, 11 Jul 2025 09:03:31 GMT
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
```

```
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
X-Cache: Miss from cloudfront
Via: 1.1 043fc2faaa02eeb59193e3fa300adb6a.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: AMS1-C1
Alt-Svc: h3=":443"; ma=86400
X-Amz-Cf-Id: nvhYbHvbnIq1Aa5E_HxvWRsG7ug3uhYAu4NpJRJSvXATe-S01a02Hg==

{"code":400,"message":"Limit of retry exceed, wait or use previous email","data":{"timeToWait":228}}
```

User enumeration can be automated using Burp Suite's Intruder tool. In testing, 1,000 requests were sent in 7 seconds using the same captchaToken, demonstrating the lack of effective rate limiting and CAPTCHA validation.

Results

Positions

▼ Capture filter: Capturing all items

▼ View filter: Showing all items

| Request ▼ | Payload | Status code | Response received |
|-----------|---------|-------------|-------------------|
| 1002      | 1001    | 200         | 58                |
| 1001      | yer1    | 200         | 145               |
| 1000      | 1000    | 200         | 55                |
| 999       | 999     | 200         | 57                |
| 998       | 998     | 200         | 58                |
| 997       | 997     | 200         | 55                |
| 996       | 996     | 200         | 54                |

Request

Response

Pretty

Raw

Hex

Render

1 HTTP/2 200 OK

2 Content-Type: application/json

3 Date: Fri, 11 Jul 2025 09:12:59 GMT

4 Vary: Origin

5 Vary: Access-Control-Request-Method

6 Vary: Access-Control-Request-Headers

7 Access-Control-Allow-Origin: \*

8 X-Content-Type-Options: nosniff

9 X-Xss-Protection: 0

10 Cache-Control: no-cache, no-store, max-age=0, must-revalidate

11 Pragma: no-cache

12 Expires: 0

13 Strict-Transport-Security: max-age=31536000 ; includeSubDomains

14 X-Frame-Options: DENY

15 X-Cache: Miss from cloudfront

16 Via: 1.1 e328b143eb69c36369a2def78300d502.cloudfront.net (CloudFront)

17 X-Amz-Cf-Pop: AMS1-C1

18 Alt-Svc: h3=":443"; ma=86400

19 X-Amz-Cf-Id: 0ieZzr12LQ2LIHicvZeFjhv4mudv05Jjm6ru2LX6g\_zwdNXBday8TQ==

20

21 {

"code":400,

"message":"Limit of retry exceed, wait or use previous email",

"data":{

"timeToWait":232

}

}

### 3.1.1.2 Signup

The application discloses that the email is already in use.

#### HTTP request:

```
POST /api/v1/public/security/auth/signup HTTP/2
Host: api.demobroadway.com
Content-Length: 328
```

```
User-Language: en
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://www.demobroadway.com
Referer: https://www.demobroadway.com/
Accept-Encoding: gzip, deflate, br

{"email":"1231@gmail.com","password":"07e4b2df39aad678936046bd645d92c5e9061b2ecdf68973d3
b94835228d8029","isAgreeWithConditions":true,"receivePromotions":false,"defaultWalletCoin":
"SGD","isTouchDevice":false,"language":"en","currency":"USD","depositModalAppearance"
:{ "enabled":false,"timeZone":"Europe/Kiev","captchaToken":"" }}
```

#### HTTP response:

```
HTTP/2 200 OK
Content-Type: application/json
Date: Tue, 15 Jul 2025 09:27:36 GMT
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Cache: Miss from cloudfront
Via: 1.1 098d6395a0558ff140166a3bcc78ccbe.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: WAW51-P2
Alt-Svc: h3=":443"; ma=86400
X-Amz-Cf-Id: M-pDs73M8i0jyuuQW_AKjhtZ6yitgqS6kQ8jZ_DwDNkG9WGM36040Q==

{"code":400,"message":"Email already in use"}
```

#### Recommendations:

Provide less verbose responses in the functionality. The server should return the same generic messages regardless if the username/email address exists or not. A message such as 'Further instructions have been sent to your email address' or similar.

#### References:

- <https://cwe.mitre.org/data/definitions/204.html>
- [https://cheatsheetseries.owasp.org/cheatsheets/Authentication\\_Cheat\\_Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html)
- <https://www.rapid7.com/blog/post/2017/06/15/about-user-enumeration/>

### 3.1.2 Improper Captcha Implementation

Severity: **Low**

Location:

- <https://api.demobroadway.com/api/v1/public/security/auth/signup>
- <https://api.demobroadway.com/api/v1/public/security/auth/signin>
- <https://api.demobroadway.com/api/v1/public/security/auth/2fa/signin>

Impact:

Without proper CAPTCHA validation on the login, signup, and password reset endpoints, the application is vulnerable to automated attacks such as credential stuffing, mass account creation, and abuse of the password reset functionality.

Vulnerability Details:

CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) is intended to distinguish human users from automated scripts by requiring interaction that is easy for humans but difficult for bots. When not properly validated server-side, it fails to provide any real protection against automated abuse.

The application does not validate the captchaToken parameter on the login, signup, and password reset endpoints. Although the parameter is present, the server only checks that it is not empty, without verifying its authenticity.

Proof of Vulnerability:

The application does not validate the actual value of the **captchaToken** parameter.

As demonstrated in the request below, the application processes the signup request successfully even when the captchaToken is replaced with arbitrary characters. The only verification performed is that the captchaToken is not empty.

#### HTTP request:

```
POST /api/v1/public/security/auth/signup HTTP/2
Host: api.demobroadway.com
Content-Length: 328
User-Language: en
Accept-Language: en-US,en;q=0.9
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Accept: application/json, text/plain, */*
Content-Type: application/json
Origin: https://www.demobroadway.com
Referer: https://www.demobroadway.com/
Accept-Encoding: gzip, deflate, br

{"email":"1231@gmail.com","password":"07e4b2df39aad678936046bd645d92c5e9061b2ecdf68973d3b94835228d8029","isAgreeWithConditions":true,"receivePromotions":false,"defaultWalletCoin":"SGD","isTouchDevice":false,"language":"en","currency":"USD","depositModalAppearance":{"enabled":false},"timeZone":"Europe/Kiev","captchaToken":" :) "}
```

#### HTTP response:

```
HTTP/2 200 OK
Content-Type: application/json
Date: Mon, 14 Jul 2025 12:07:58 GMT
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
X-Cache: Miss from cloudfront
Via: 1.1 e10153740ff95eb4d0c9f3172baeb43e.cloudfront.net (CloudFront)
X-Amz-Cf-Pop: AMS1-C1
Alt-Svc: h3=":443"; ma=86400
X-Amz-Cf-Id: UW4HFjLS4Dm_N8-h_hjwKKRpIN9Yoy48dwBpEDizlgQYiv85QF2V5g==

{"username":"Dygdxdxijnd","token":"ee06cfa2b2574f37a7792844cbc9a4683291dad77f235050ba108f610d7b17ad","uuid":"dad416f-650b-43e5-9d5c-22b0878d6f90","isTfaEnabled":false}
```

### Recommendations:

It is recommended to:

- Implement proper server-side validation of the **captchaToken** parameter for all critical endpoints, including login, signup, and password reset.
- Ensure the CAPTCHA is verified using the official CAPTCHA provider's API (e.g., Google reCAPTCHA) and that only valid, unexpired tokens are accepted before processing the request.

### References:

- <https://cwe.mitre.org/data/definitions/358.html>
- <https://developers.google.com/recaptcha/docs/verify>

## 3.2 Informational severity findings

### 3.2.1 CORS misconfiguration

Severity: **Informational**

Location:

- <https://api.demobroadway.com/>

Impact:

A CORS misconfiguration can leave the application at a high-risk of compromise resulting in an impact on the confidentiality and integrity of data by allowing third-party sites to carry out privileged requests through website's authenticated users such as retrieving user setting information or personal user's data.

Vulnerability Details:

Cross-origin resource sharing (CORS) is a browser mechanism which enables controlled access to resources located outside of a given domain. It extends and adds flexibility to the same-origin policy (SOP). However, it also provides potential for cross-domain based attacks, if a website's CORS policy is poorly configured and implemented. CORS is not a protection against cross-origin attacks such as cross-site request forgery (CSRF).

Proof of Vulnerability:

#### HTTP request:

```
OPTIONS /api/v1/public/security/auth/signin HTTP/2
Host: api.demobroadway.com
Accept: */*
Access-Control-Request-Method: POST
Access-Control-Request-Headers: content-type,user-language
Origin: https://evil.com
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/138.0.0.0 Safari/537.36
Referer: https://www.demobroadway.com/
Accept-Encoding: gzip, deflate, br
Accept-Language: en-US,en;q=0.9
```

#### HTTP response:

```
HTTP/2 200 OK
Date: Wed, 16 Jul 2025 13:16:58 GMT
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Access-Control-Allow-Origin: *
Access-Control-Allow-Methods: GET,POST,OPTIONS,DELETE,PUT,PATCH
Access-Control-Allow-Headers: content-type, user-language
```

```
Access-Control-Max-Age: 1800
Allow: GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE, PATCH
X-Content-Type-Options: nosniff
X-Xss-Protection: 0
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
```

### Recommendations:

CORS vulnerabilities typically stem from misconfigurations, making it essential to follow secure practices to prevent exploitation.

When dealing with web resources that contain sensitive information, the 'Access-Control-Allow-Origin' header should explicitly specify a trusted origin.

Only trusted sites should be included in this header, and dynamically reflecting origins from cross-domain requests without proper validation should be strictly avoided, as it poses a significant security risk. Additionally, the use of 'Access-Control-Allow-Origin: null' is discouraged, as it can be misused in certain contexts. **Wildcards should also be avoided, particularly within internal networks, to minimize unnecessary exposure and potential attack vectors.**

### References:

- [https://portswigger.net/kb/issues/00200600\\_cross-origin-resource-sharing](https://portswigger.net/kb/issues/00200600_cross-origin-resource-sharing)
- <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

### 3.2.2 Weak password policies

Severity: **Informational**

Location:

- <https://api.demobroadway.com/api/v1/public/player/change-password>

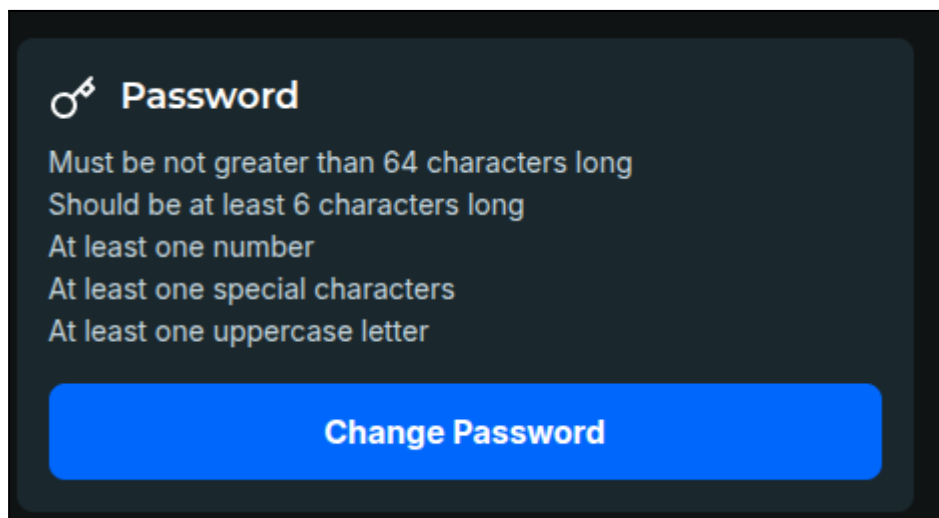
Impact:

A weak password policy increases the probability of an attacker having success using brute force and dictionary attacks against user accounts. An attacker who can determine user passwords can take over a user's account and potentially access sensitive data.

Vulnerability Details:

During testing, it was discovered that the users within the environment can set up weak passwords that could be easily cracked with brute force and dictionary attacks. This allows attackers to gain unauthorized access to sensitive resources and data within the network, compromising the confidentiality, integrity, and availability of the information.

Proof of Vulnerability:



Recommendations:

- Prohibit the creation of passwords less than 8 characters in length.

References:

- <https://cwe.mitre.org/data/definitions/521.html>
- <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf>

## APPENDIX A - Performed tests according to OWASP Top 10

| #   | Vulnerability                                     | Description                                                                                                                                                                                                                                                                                                                                                                                                   | Status        |
|-----|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|
| A01 | <b>Broken Access Control</b>                      | Access controls enforce policies so that users cannot act outside of their intended permissions. Failures typically lead to unauthorized information disclosure or modification, destruction of data, or performing a business function outside the user's limits.                                                                                                                                            | <b>Unsafe</b> |
| A02 | <b>Cryptographic Failures</b>                     | Cryptographic Failures involve protecting data in transit and at rest. This includes passwords, credit card numbers, health records, personal information, and business secrets that require extra protection, especially if that data falls under privacy laws such as GDPR or regulations like PCI Data Security Standard (PCI DSS) for financial data.                                                     | <b>Safe</b>   |
| A03 | <b>Injection</b>                                  | An application is at risk when user-supplied data is not validated, filtered, or sanitized by the application; dynamic queries or non-parameterized calls without context-aware escaping are used directly in the interpreter; hostile data is used within object-relational mapping (ORM) search parameters to extract additional, sensitive records; or when hostile data is directly used or concatenated. | <b>Safe</b>   |
| A04 | <b>Insecure Design</b>                            | According to OWASP, "Secure design is a culture and methodology that constantly evaluates threats and ensures that code is robustly designed and tested to prevent known attack methods. Secure design requires a secure development lifecycle, some form of secure design pattern or paved road component library or tooling, and threat modeling."                                                          | <b>Safe</b>   |
| A05 | <b>Security Misconfiguration</b>                  | This category includes such things as missing security hardening across any part of the application stack, improperly configured permissions on cloud services, any unnecessary features that are enabled or installed, and unchanged default accounts or passwords. The former category XML External Entities (XXE) is now included in Security Misconfiguration.                                            | <b>Unsafe</b> |
| A06 | <b>Vulnerable and Outdated Components</b>         | This category includes any software that is vulnerable, unsupported, or out of date. If you do not know the versions of your components – including all direct and indirect dependencies – or you do not regularly scan and test your components, you are likely at risk.                                                                                                                                     | <b>Safe</b>   |
| A07 | <b>Identification and Authentication Failures</b> | Security risk occurs when a user's identity, authentication, or session management is not properly handled, allowing attackers to exploit passwords, keys, session tokens, or implementation flaws to assume users' identities temporarily or permanently.                                                                                                                                                    | <b>Unsafe</b> |

|            |                                                 |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |             |
|------------|-------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
| <b>A08</b> | <b>Software and Data Integrity Failures</b>     | This includes software updates, critical data, and CI/CD pipelines that are implemented without verification. An example of this includes objects or data encoded or serialized into a structure that an attacker can modify. Another example is an application that relies upon plugins, libraries, or modules from untrusted sources. Insecure CI/CD pipelines that can introduce the potential for unauthorized access, malicious code, or system compromise also fit into this category. Lastly, applications with auto-update functionality, in which updates are downloaded without sufficient integrity verification and applied to a previously trusted application, are considered software and data integrity failures because attackers could infiltrate the supply chain to distribute their own malicious updates. | <b>Safe</b> |
| <b>A09</b> | <b>Security Logging and Monitoring Failures</b> | This category includes errors in detecting, escalating, and responding to active breaches. Without logging and monitoring, breaches cannot be detected. Examples of insufficient logging, detection, and monitoring include not logging auditable events like logins or failed logins, warnings and errors that generate inadequate or unclear log messages, or logs that are only stored locally. Failures in this category impact visibility, incident alerting, and forensics.                                                                                                                                                                                                                                                                                                                                               | <b>Safe</b> |
| <b>A10</b> | <b>Server-Side Request Forgery</b>              | Server-Side Request Forgery occurs when a web application fetches a remote resource without validating the user-supplied URL. An attacker can coerce the application to send a crafted request to an unexpected destination, even when protected by a firewall, VPN, or another type of network ACL. Though SSRF shows a relatively low incidence rate in the data OWASP reviewed, this category was added based on the industry survey results; users are concerned that SSRF attacks are becoming more prevalent and potentially more severe due to increased use of cloud services and the complexity of architectures.                                                                                                                                                                                                      | <b>Safe</b> |