

Pentest Scenario Solutions

Introduction

The pentest scenario in this case is NOT jeopardy style. This is a real life engagement scenario, so it should be approached as such. It aims to showcase how a real life pentests are typically approached by professionals and how to cancel out excessive noise the applications tend to have in real life. That is why you start with a statement of work to analyze the scope and limitations. That said, let's dive straight into the solution. All the questions presented as challenges are designed to guide you through the scenario, not get you stuck.

Note that all the "challenges" expecting answers were designed to be case insensitive, so it doesn't matter how you will provide the right answer. (security or SeCURitY, both will be correct)

SoW – Statement of Work

A statement of work is a document that is created between two companies – the supplier and the customer, which will provide services. The supplier in this case provides a service defined by the Statement of Work to the customer. A statement of work is considered a legal document and as such a breach of contract could result in fines or legal repercussions. That's why the statement of work should contain clearly defined expectations and limitations in case they are applicable.

The statement of work provided in the event was "SoW – Web App Pentesting 2025.docx"



BLIND SECURITY

Statement of Work

Blind Security ("Customer")

Kaykasou 19, Aristotelous
54044 Thessaloniki
Greece

Email: accounts@blindsecurity.net
Telephone: 240765432
VAT: 589552658

Contact: Petros Papadopoulos

CSCGR Pentesting Team ("Supplier")

Athens University of West Attica
122 43
Greece

Email: your@team.email

VAT: N/A

Contact: [Your-Team-Lead-Name](#)

Solutions

SoW Limitations:

Question: Based on the Statement of Work, which type of attack is strictly prohibited that could crash the website?

Answer: Denial of Service

Explanation: Based on the statement of work under the Scope Limitations clause, 4 types of attacks are defined as out of scope. Phishing Attacks, Denial of Service & Distributed Denial of Service, Active scans and Password Guessing/Password Bruteforcing.

While the last two attacks could have an impact on the availability of the service, the typical attack that could crash a website would be Denial of Service.

Scope Limitations

No user accounts will be provided for any of the aforementioned application. Therefore this can be approached as a black-box assessment.

Additionally, the following attacks are deemed out of scope and therefore are not permitted:

1. Phishing Attacks / Social Engineering Attacks
2. Denial of Service Attacks (DoS) and Distributed Denial of Services Attacks (DDoS)
3. Active scans on the websites in scope (dirbuster, gobuster, sqlmap, etc)
4. Password Guessing / Password Bruteforcing

All other attacks not specified by Blind Security are permitted.

SoW Timeframe

Question: Based on the Statement of Work, how long will the engagement last?

Answer: 6 hours

Explanation: Similarly to the above, by reviewing the provided Statement of Work under the “Execution Dates” and “Location” clauses, you can find the time constraints of the assessment.

Execution Dates
<i>The assessment is planned to commence on 28th of March 2025 at 13:30 Athens time and completed on 28th of March 2025 at 19:30 Athens time.</i>

Location

The work can be performed from either Athen's main event location (University of West Attica), Thessaloniki's hub location (University of Macedonia), Patra's hub location (University of Peloponnese), Samos' hub location (University of Aegean), or remote. The durations are as shown below:

Site	Duration
Office Locations	6 hours
Remote Location	6 hours

(And as you can tell from the screenshot above, the locations were not properly updated to reflect the proper site for Thessaloniki)

Technology Stack

Question: What technology stack is the backend of the portal application based on?

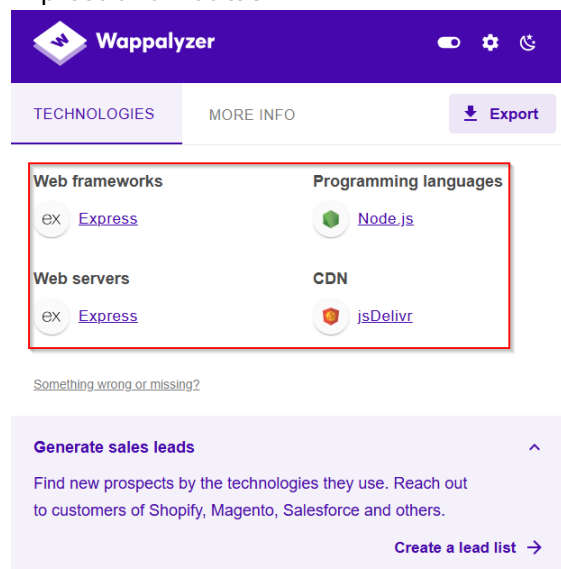
Answer: NodeJS

Explanation: It's now time to start assessing the application in scope. (The application's URL was found at the "Scope" clause of the Statement of Work.

The URL for onsite participants was: "portal-dev.blindsecurity.net".

By navigating to the URL and using a utility such as "Wappalyzer" we get some options:

Express and NodeJS



The runtime environment is NodeJS which answers the question above as the application is based on the runtime environment it executes on. Express on the other hand is used to manage the application's dynamic features, hence the Web Framework. The "Web Servers" reading is actually

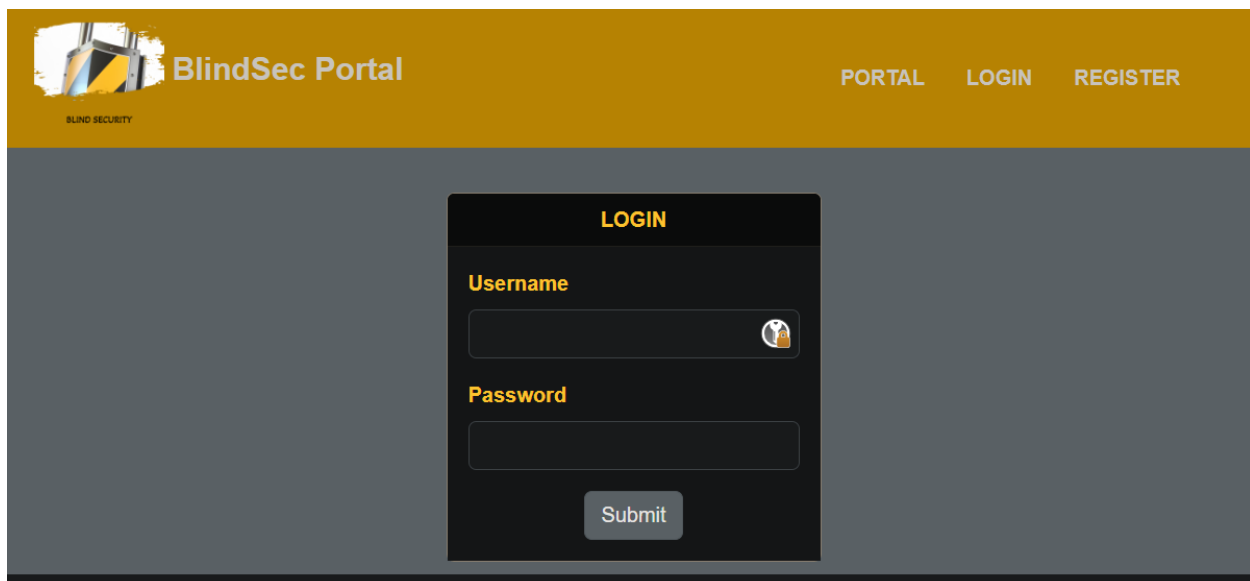
wrong as it directly reads the “Server” header of the HTTP response received by the server which says “Express”. That’s there to mislead in this case.

Checkpoint

Obviously we cannot answer further questions without exploring and identifying some other issues in the application itself. Therefore let’s take a brief moment to go through the application here and we will get back to the questions as soon as we identify and exploit the application’s vulnerabilities.

Exploitation

Opening the URL, the first thing we are greeted with is a login page. This means we cannot access anything behind the login page – or can we?

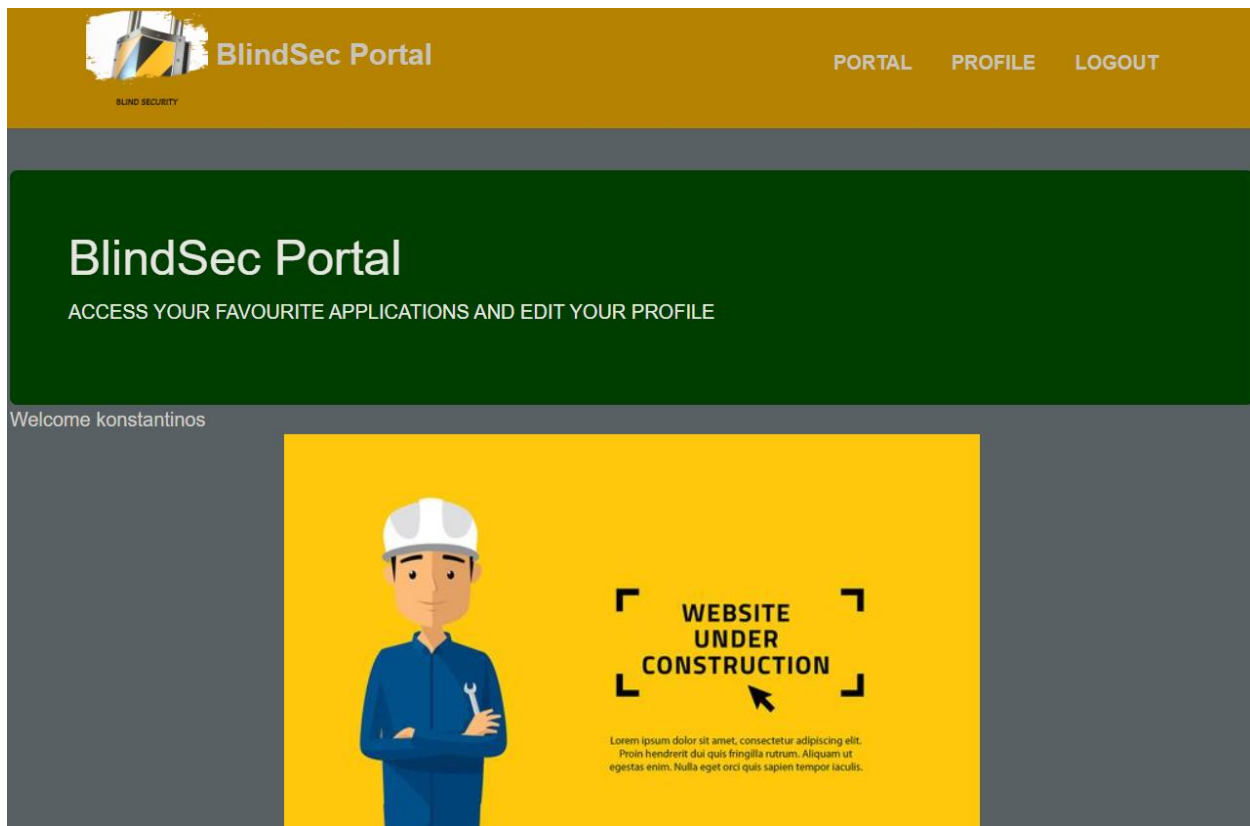


At the very top right of the screen there is a “Register” option for the application. It’s a bit weird to have a “Register” option for an application which is designed to be used only by internal employees as a “jumpbox” to different internal applications. Maybe this is an internal feature which will be disabled in production or a mistake on the business logic the developers made.

In any case, that’s something we can utilize to create an account and proceed testing the internals and logic of the application. Let’s create an account.

First thing we notice is that there is no password policy enforced. Therefore we are allowed to enter any insecure password we wish and of any length. That’s already a finding, but not exploitable directly as we are not allowed to perform “Password Guessing” or “Password Bruteforcing” attacks.

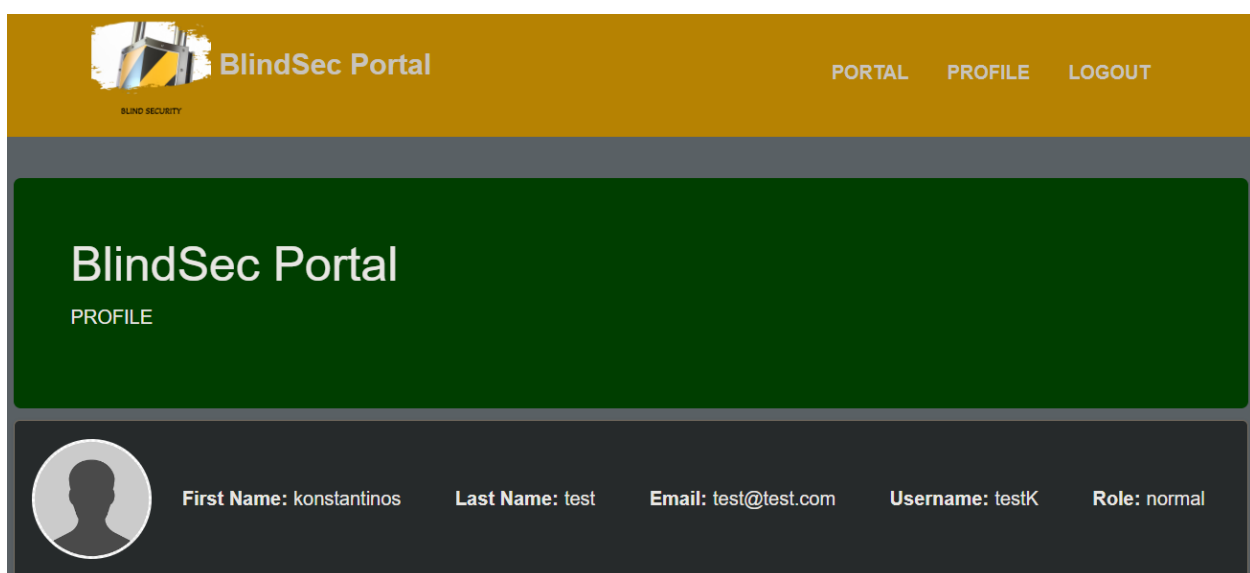
After creating the account we are required to setup MFA. That’s a nice addition for the developers, they’re definitely compliant with some policies. Let’s setup the MFA and access the application.



Apparently the application is still under construction and there's no real features that are implemented yet. First thing I'd like to do is to go through all the available interfaces in case there's something else we can access that's well implemented.

The "Portal" endpoint is the main page which we see above.

The "Profile" endpoint on the other hand contains information about our user as shown below.



That said, some functionality is available. Let's further explore the application through common known files. Let's try robots.txt and sitemap.xml to see if there are any interesting artifacts there we could use.





Indeed some entries exist in robots.txt as shown below:

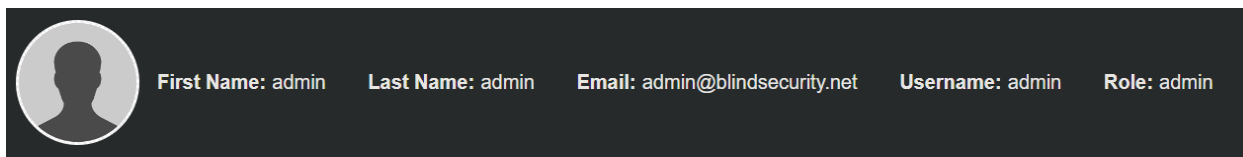
```
User-agent: *
Disallow: /public
Disallow: /test
Disallow: /MFA_Enroll
Disallow: /admin
Disallow: /panel
Disallow: /users
Disallow: /portal
Disallow: /feedback
Disallow: /Y3NjezBoX24wX3IwYjB0c180c_jNfczNuczF0MYzPz99
```

There are various different endpoints here which can indicate existing/to be implemented functionality. Let's try them out one by one.

By base64 decoding the last entry, we get a flag. (1)

The only endpoint that works – which we don't currently have is /users. Apparently with /users we can have an overview of all the registered users in the platform. This should be an admin functionality and not everyone should be able to access it, therefore it seems like a Broken Access Control issue. We can see different roles as well “normal” and “admin”, so indeed this is a Broken Access Control issue.

	First Name: test	Last Name: test	Email: blindsecurity.composure370@simplelogin.com	Username: papaya	Role: normal
	First Name: Maria	Last Name: Giannakaki	Email: maria.giannakaki@blindsecurity.net	Username: maria	Role: normal
	First Name: test	Last Name: test	Email: test123@gmail.com	Username: test	Role: normal
	First Name: Christos	Last Name: Markatos	Email: christos.markatos@blindsecurity.net	Username: chris	Role: admin

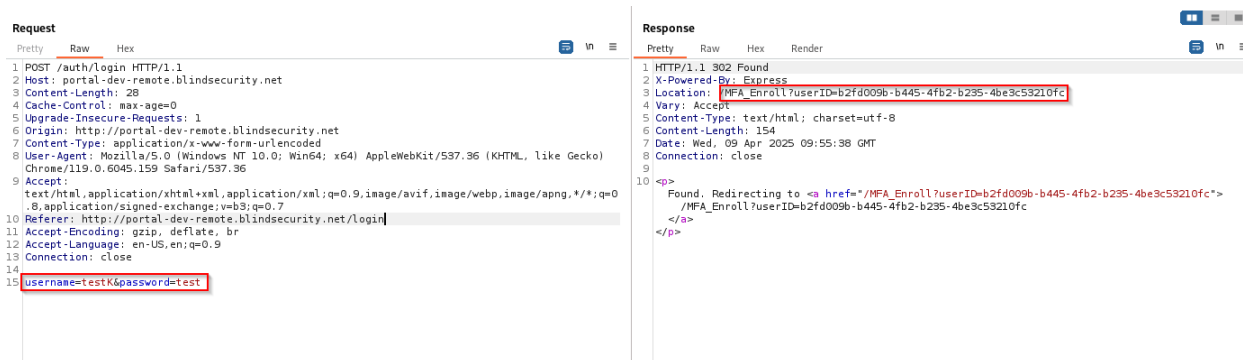


So we have 2 admins and several users.

Since that's all the functionality we've got available to us for testing, let's have a look and dissect the login functionality. (Only available flow within the application at this point of testing)

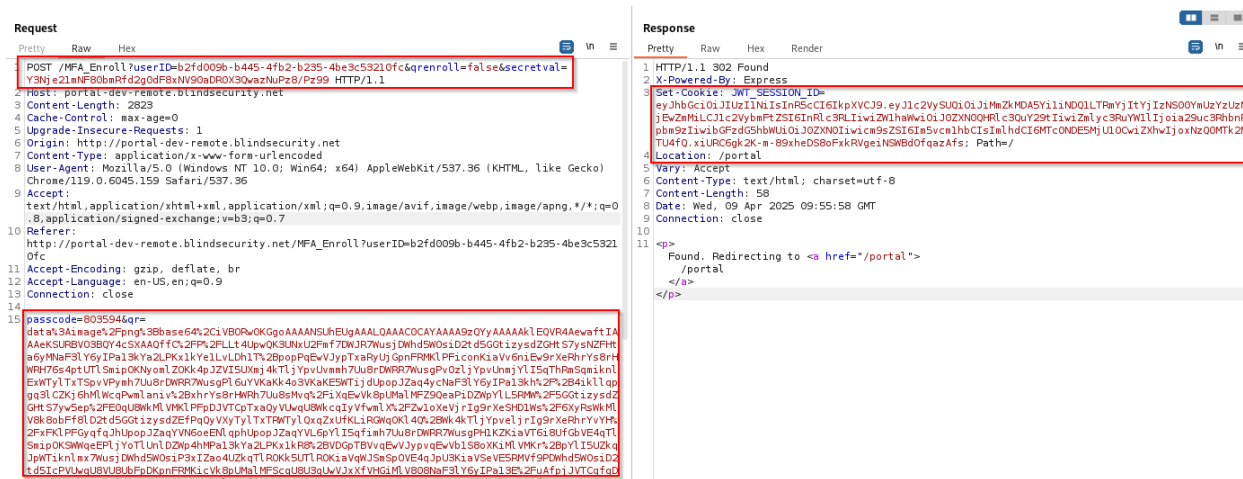
We will have to proxy all our traffic through Burp if we want to be able to dissect all the calls of the login process. Let's do that.

The login flow is comprised of 2 steps from the user's perspective. It requests the user their credentials and then proceeds by requesting to verify their identity over MFA. This seems solid enough. Let's dissect this by reviewing the exact Request-Response pairs of the communication.



Indeed the first Request-Response pair takes as input the username and password of the user to log in for verification. It then redirects the user to the “/MFA_Enroll” endpoint with the identified userID, which is probably our user's ID.

Let's have a closer look at the Request-Response pair of the “/MFA_Enroll” endpoint as well.



As you can see there are various points of interest in this pair. We can see the parameters being passed in the POST request (userID, qrenroll and secretval) as well as the body of the request containing the passcode (which makes sense) and another argument called “qr” which is interesting. The response is a typical authentication success response provided by an application, it sets the session token and redirects to the main page of the application. Let’s have a closer look at these parameters.

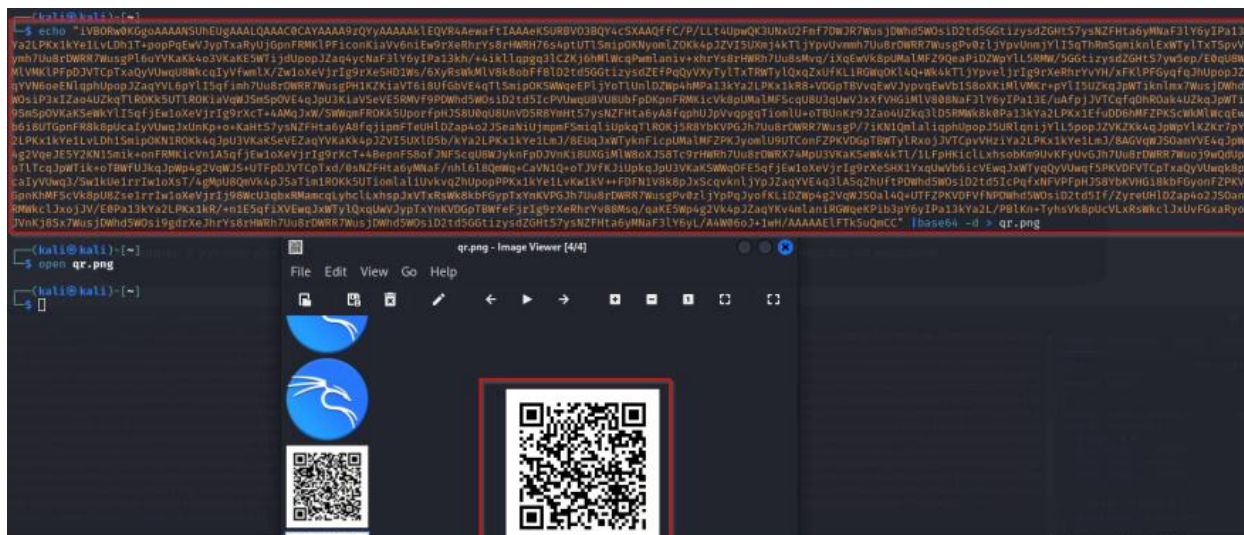
userID: As previously, specifies the right userID to log in to.

qrEnroll: Probably specifies whether this is the first time a user logs in and if they should set up their MFA.

secretVal: base64 encoded value. Decodes to a flag (2).

passcode: The MFA token the user added as input.

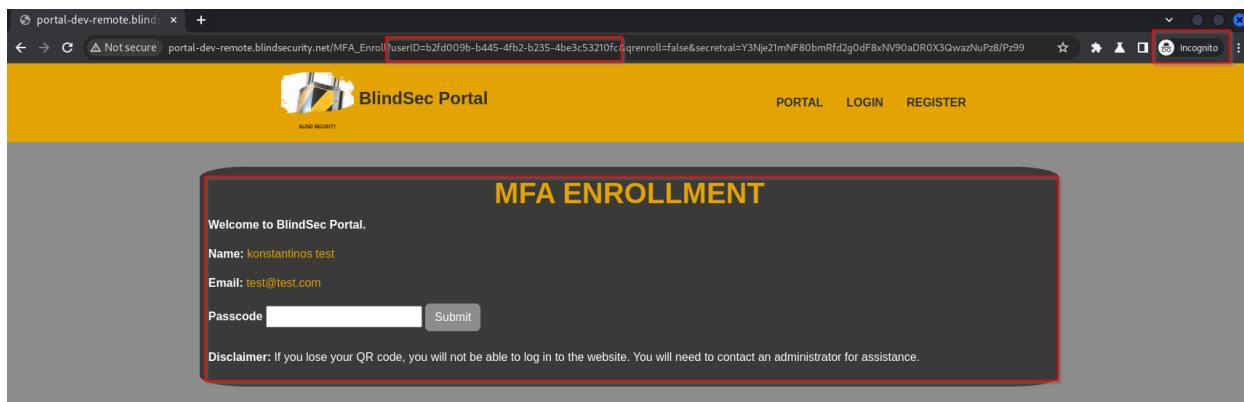
qr: Once base64 decoded, it appears to be the literal png of the QR code of the user requested to login with as shown below. (Why?)



Since we can intercept the QR code of any user we can easily bypass MFA due to the improper implementation the developers performed on this endpoint. Secrets such as QR codes should never be exposed to the client side after the initial setup of the MFA.

However, it seems we still need to obtain the victim’s credentials to end up in the MFA endpoint in order to bypass the MFA. What if we didn’t though? Let’s try to directly access via incognito mode the “/MFA_enroll” endpoint by selecting our known userID. Would we be able to provide the MFA token for our user and bypass it?

We can indeed directly access this endpoint and go ahead and add the right passcode since we can bypass MFA to the user as shown below:



This means that if we can somehow guess the userID of other users, we can perform complete account takeover to all users within the platform. But we have a limitation. The userID here is UUIDv4, which is unguessable.

A Version 4 UUID is a universally unique identifier that is generated using random numbers. The Version 4 UUIDs produced by this site were generated using a secure random number generator.

Given the above, we need to have some way of identifying valid userIDs that actually match some user in the platform. If only there was a way,...

Let's circle back to the Broken Access Control vulnerability and analyze it a bit further. (/users endpoint)

```

<div class="user-container"> (flex)
  
  <input type="hidden" id="userID" value="29116f0c-de3f-4b5a-9b9b-a0591048a3c8">
  <div class="user-field"> (flex)
    <label for="firstname">First Name:</label>
    <span id="firstname">Athenasia</span>
  </div>
  <div class="user-field"> (flex)
    <label for="lastname">Last Name:</label>
    <span id="lastname">Psomiadou</span>
  </div>
  <div class="user-field"> (flex) == $0
    <label for="email">Email:</label>
    <span id="email">athanasia.psomiadou@blindsecurity.net</span>
  </div>

```

In the source code of the page we can see that alongside the details of the users there's also their userID. So we do have access to the userID. Let's locate a high value target and attempt to take over their account.

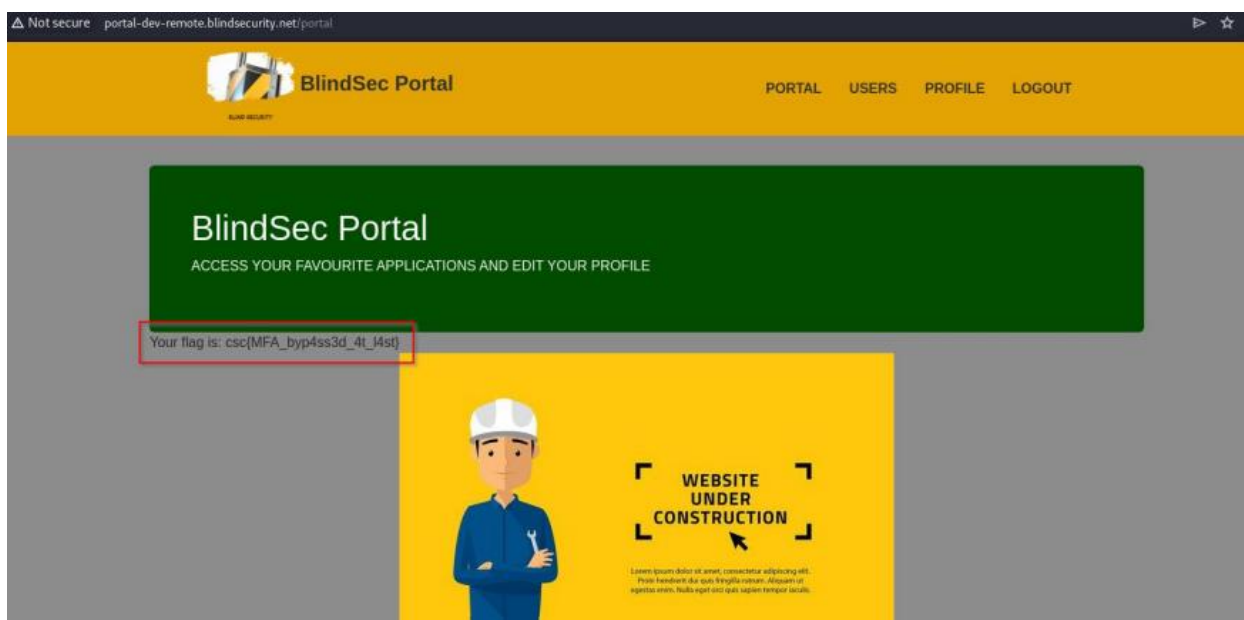
An interesting high value target is the built-in admin user. Now that we know their userID we can skip the first step of the authentication process (since we do not know their password) and then hijack their QR code to bypass their MFA. That's account takeover.

```

<div class="user-container"> (flex)
  
  <input type="hidden" id="userID" value="fbb325e8-50da-435f-9b88-a7d15dad15b2">
  <div class="user-field"> (flex)
    <label for="firstname">First Name:</label>
    <span id="firstname">admin</span>
  </div>
  <div class="user-field"> (flex)
    <label for="lastname">Last Name:</label>
    <span id="lastname">admin</span>
  </div>
  <div class="user-field"> (flex)
    <label for="email">Email:</label>
    <span id="email">admin@blindsecurity.net</span>
  </div>
  <div class="user-field"> (flex) == $0
    <label for="username">Username:</label>
    <span id="username">admin</span>
  </div>

```

Indeed we did it.



That's another flag (3)

Time to go back to the questions.

OWASP Welcome

Question: What is the OWASP Top 10 identifier of the "Welcome" flag vulnerability you found?

Answer: A07:2021

Explanation: Once you have found the Welcome and WhoAmI flags you can answer this question. The welcome flag was related to the initial flag we identified in the "/MFA_Enroll" endpoint. Let's check the OWASP Top 10 vulnerabilities.

The latest one at the time of writing is Top 10 2021.

2021

A01:2021-Broken Access Control

A02:2021-Cryptographic Failures

A03:2021-Injection

A04:2021-Insecure Design

A05:2021-Security Misconfiguration

A06:2021-Vulnerable and Outdated Components

A07:2021-Identification and Authentication Failures

A08:2021-Software and Data Integrity Failures

A09:2021-Security Logging and Monitoring Failures*

A10:2021-Server-Side Request Forgery (SSRF)*

Since this issue is related to the MFA implementation, it impacts the authentication part of the application. Even though the right answer could be easily mistaken with “A05:2021” Security Misconfiguration, it’s not the right one. Have a look at the explanation of that control:

A05:2021-Security Misconfiguration moves up from #6 in the previous edition; 90% of applications were tested for some form of misconfiguration. With more shifts into highly configurable software, it’s not surprising to see this category move up. The former category for XML External Entities (XXE) is now part of this category.

The security misconfiguration is a broader category of controls. We always measure categories by impact and since this issue heavily impacts the authentication (Leads to Broken Authentication in combination with the fact we can completely bypass the first step of the authentication) the correct answer is A07:2021 – Identification and Authentication Failures

OWASP WhoAml

Question: What is the OWASP Top 10 identifier of the "WhoAml" flag vulnerability vulnerability you found?

Answer: A01:2021

Explanation: This issue affects the “/users/” endpoint. This was a Broken Access Control as only the admin user should be able to view this page, but all users can as it does not enforce proper authorization checks on the role of each user. Therefore it’s A01:2021 – Broken Access Control as shown in the above picture.

Maximum Impact

Question: What attack can you perform to showcase maximum impact to the website?

Answer: Account Takeover

Explanation: As we noticed during the pentest we can completely bypass the login process of the application. However, just bypassing the authentication does not mean that’s the maximum impact. The maximum impact is exactly what is the worst that could happen. In this scenario, we

can perform account takeover (ATO) on any registered user on the application from a black-box perspective.

Executive Summary

Question: Which type of people is the Executive Summary part of a report addressed to?

Answer: stakeholders

Explanation: Managers and Single Points of Contact (SPoC) members from clients as well as C-level executives such as Chief Information Security Officer (CISO) or Chief Security Officer (CSO), etc are all addressed as “Stakeholders” of the assessment. This is official terminology.

Admin

Question: Who is the other admin in the website?

Answer: chris

Explanation: The answer here is either “chris” (Username) or “christos markatos” (Full Name). Both of these can be found at the “/users/” endpoint.

Bug Bounty

Question: By which file can you tell that a website is open for bug bounties?

Answer: security.txt

Explanation: Companies that are open to bug bounties and encourage them, typically provide pentesters a way to contact them via a secure channel for Responsible Disclosure procedures. This is typically explained in a known location within the website. That is “/.well-known/security.txt”

```
Hello Ethical Hacker,  
  
Thank you for having a close look on our portal. Please submit any findings you may have found on the following email address. We will forward your report through our internal procedure and get in touch with you for compensation. :)  
  
Security Officer: Katerina Morfaki  
Email: responsible-disclosure@blindsecurity.eu  
  
csc{n1c3_c4tch_m4t3!}
```

Welcome Flag

Question: First Exploitable Vulnerability

Answer: csc{mf4_4nd_wh4t_15_th4t_t0k3n?????}

Explanation: This was identified and explained in the Exploitation section when we found flag (2).

WhoAmI Flag

Question: Information disclosure issue, not exploitable alone

Answer: csc{us3r_id5_r3tri3v4bl3_br0k3n_4cc3ss_c0ntr0l}

Explanation: If you keep on searching on the “/users” endpoint you would eventually end up in the flag as shown below.

```
▼<div class="user-container"> (flex)
  
  <input type="hidden" id="userID" value="csc{us3r_id5_r3tri3v4bl3_br0k3n_4cc3ss_c0ntr0l}">
  ▼<div class="user-field"> (flex)
    <label for="firstname">First Name:</label>
    <span id="firstname">Alexandros</span>
  </div>
  ▼<div class="user-field"> (flex)
    <label for="lastname">Last Name:</label>
    <span id="lastname">Charalampidis</span>
  </div>
  ▼<div class="user-field"> (flex) == $0
    <label for="email">Email:</label>
    <span id="email">alexandros.charalampidis@blindsecurity.net</span>
  </div>
```

Maximum Impact

Question: Showcase maximum impact based on what you found!

Answer: csc{MFA_byp4ss3d_4t_l4st}

Explanation: This was identified and explained in the Exploitation section when we found flag (3)

Robots – Bonus Flag

Question: N/A

Answer: csc{0h_n0_r0b0ts_4r3_s3ns1t1v3??}

Explanation: This flag was identified in the robots.txt section on the last entry. Flag (1)

Bug Bounty – Bonus Flag

Question:

Answer: csc{n1c3_c4tch_m4t3!}

Explanation: As identified in the .well-known/security.txt entry of the application.

Reporting

Report provided in a separate PDF file.