VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0

Company: Acme Inc.

Report: VWAYYMMDD

Author: [Ahmad Mahmoud]

Date: [13-4-2023]

|  |  |
| --- | --- |
| **Broken Auth - High** | **3** |
| **XSS - High** | 4 |
| **Insecure Deserialization - high** | 5 |
| **Broken Access Control - High** | **8** |
| **Broken Access Control - High** | **9** |
| **Sensitive Data Exposure - High** | **10** |
| **Sec Misconfiguration - High** | **12** |
| **Sec Misconfiguration - High** | **14** |
| **SQLI - High** | **16** |
| **SQLI - High** | **18** |

# 

# **VWAYYMMDD## – Broken Auth - High**

**Vulnerability Exploited:** **Broken Auth using Bruteforce**

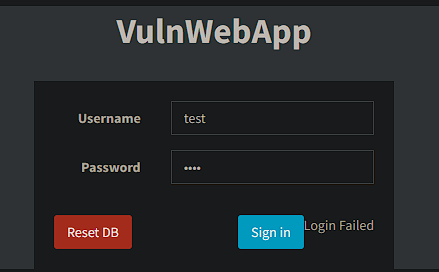
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

An unauthorized user used a brute force attack against the login form of a web application and was able to access the application.

**Vulnerability Walk-thru**:

1. First go to the login page and enter invalid credentials.
2. Go to the workspace terminal and use the “bruteforce.py” script and use this command: python bruteforce.py -U test-username.txt -P test-password.txt -f “Login Failed” http://0.0.0.0:3000/login  
   Text

   Description automatically generated
3. Enter the credentials you got from the script Graphical user interface, application, background pattern

   Description automatically generated

**Recommendations:**

We need to track fail login attempts IP address, then block the IP address if they exceed 5 fail logins for 15 minutes and also alert our SOC team of this event.  
https://cheatsheetseries.owasp.org/cheatsheets/Authentication\_Ch  
eat\_Sheet.html

# **VWAYYMMDD## – XSS - High**

**Vulnerability Exploited:** **XSS**

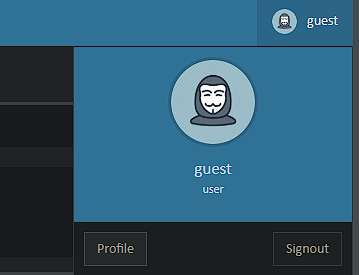
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

In the Profile Area we have input field allow you to send message to the Administrator, this input field seems to not be sanitizing for “script” tag within the messages That then  
could contain java-script code and is executable on the client  
side.

**Vulnerability Walk-thru**:

1. First go to the profile section 
2. In the input field add this payload: <script>alert(document.cookie)</script>Graphical user interface, rectangle

   Description automatically generated with medium confidence
3. When you reload the page it will response with the cookies of the userGraphical user interface, rectangle

   Description automatically generated with medium confidence

**Recommendations:**

We should use a standard library that does field sanitizing for

XSS.

https://cheatsheetseries.owasp.org/cheatsheets/Cross\_Site\_Script

ing\_Prevention\_Cheat\_Sheet.html

**VWAYYMMDD## – Insecure Deserialization - High**

**Vulnerability Exploited:** **Insecure Deserialization**

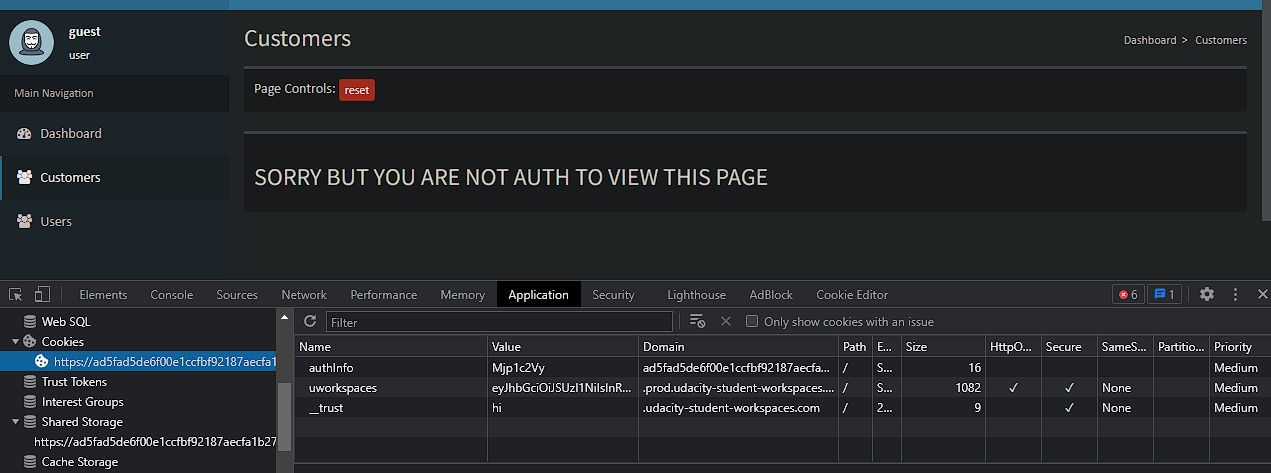
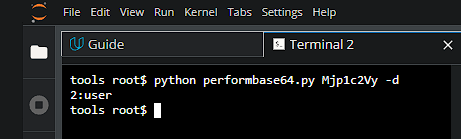
**Severity: High**

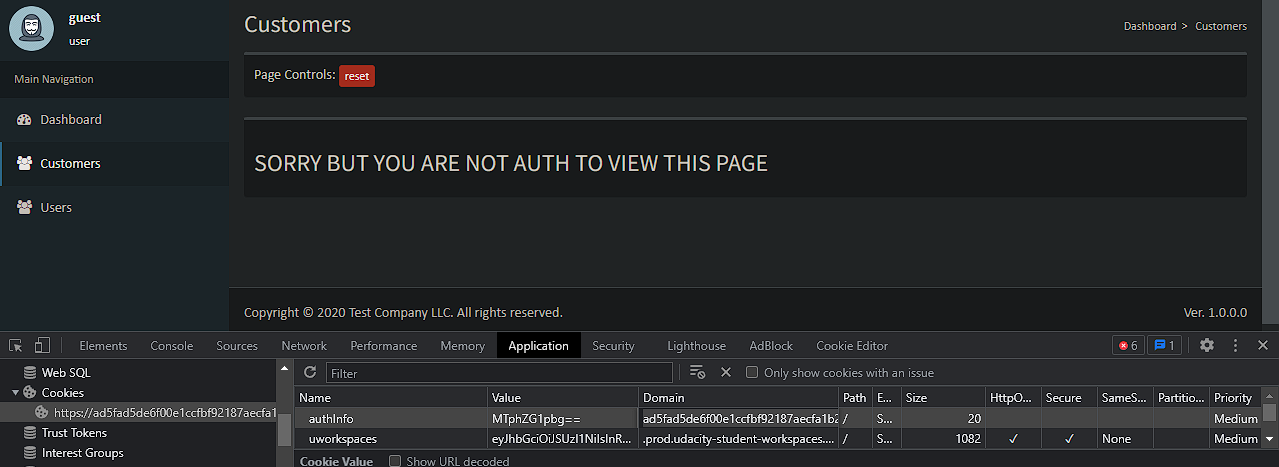
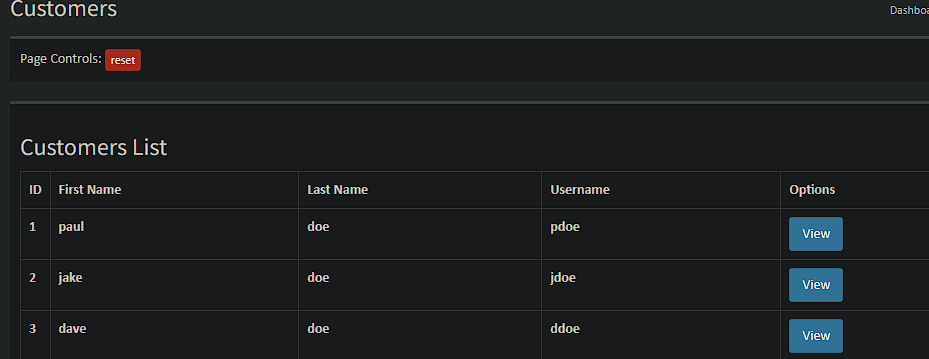
**System:** VWA Web Application

**Vulnerability Explanation**:

An insecure deserialization vulnerability was exploited, allowing the attacker to modify the cookies of normal users to match those of the admin, thereby gaining unauthorized access to private areas. This indicates a weakness in the system's handling of serialized data.

**Vulnerability Walk-thru**:

1. First go to the customer’s page while you are opening the Developers options -> application -> storage-> cookies
2. Let’s check the Authinfo value with “performbase64.py” from the workspace. 
3. Now convert 1:admin to base64 with “performbase64.py” Graphical user interface, text

   Description automatically generated
4. Then modify the Authinfo value with MTphZG1pbg== and reload the page of customers 
5. You will able to see the Customers list 

**Recommendations:**

It is recommended to implement proper input validation and sanitization measures, avoid using serialized data from untrusted sources.

<https://owasp.org/www-project-top-ten/2017/A8_2017-Insecure_Deserialization.html>

**VWAYYMMDD## –Broken Access - High**

**Vulnerability Exploited:** **Broken Access to Users**

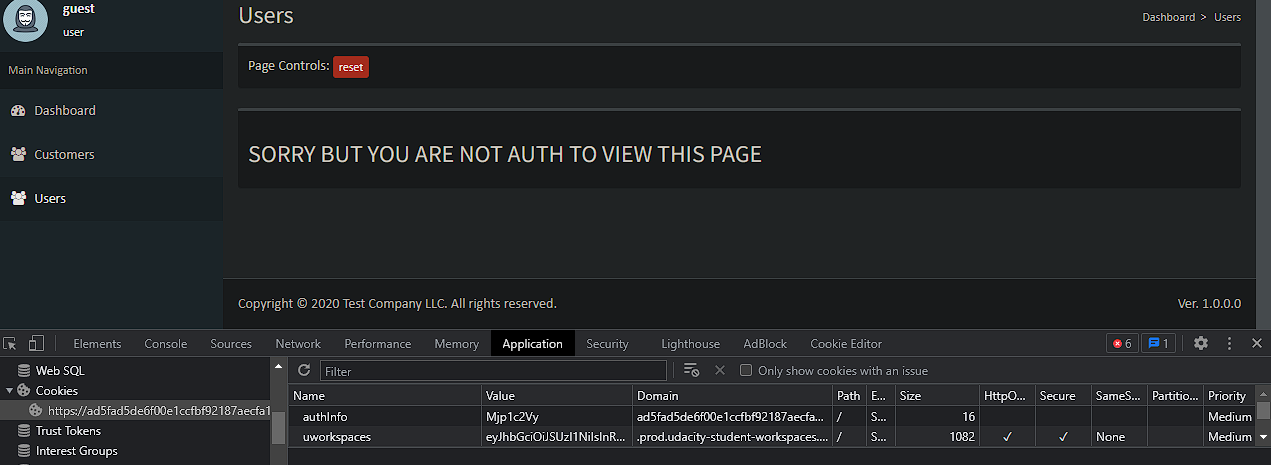
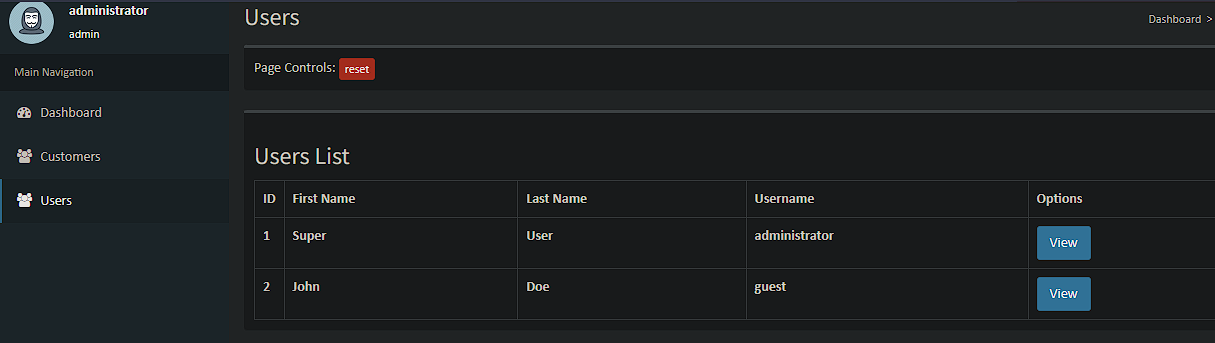
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

A broken access vulnerability was exploited, allowing the attacker to modify the cookies of Guest users to match those of the admin, there by gaining unauthorized access to the Users page.

**Vulnerability Walk-thru**:

1. First go to the Users page while you are opening the Developer options-> application -> storage -> cookies 
2. Modify the Authinfo value with “MTphZG1pbg==” then reload the page 

**Recommendations:**

it is recommended to implement a strong access control mechanism that ensures proper user authorization checks and restricts access based on roles and permissions.

<https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html>

**VWAYYMMDD## –Broken Access - High**

**Vulnerability Exploited:** **Broken Access to Customers page**

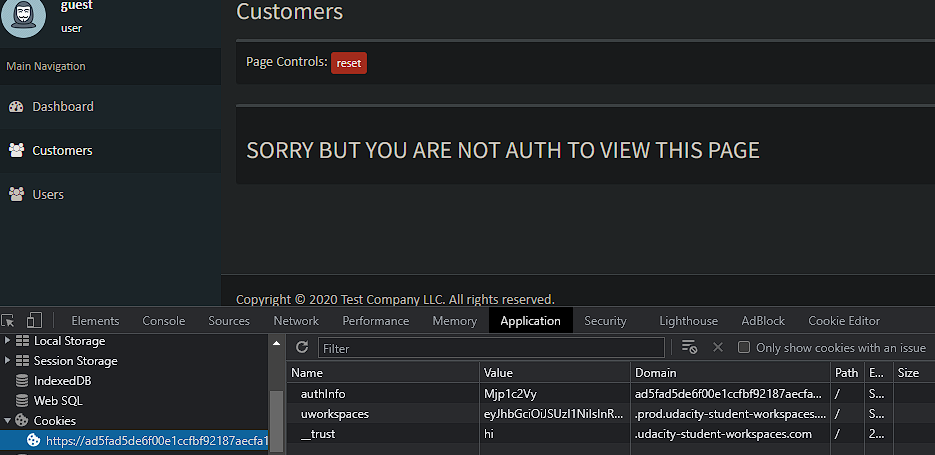
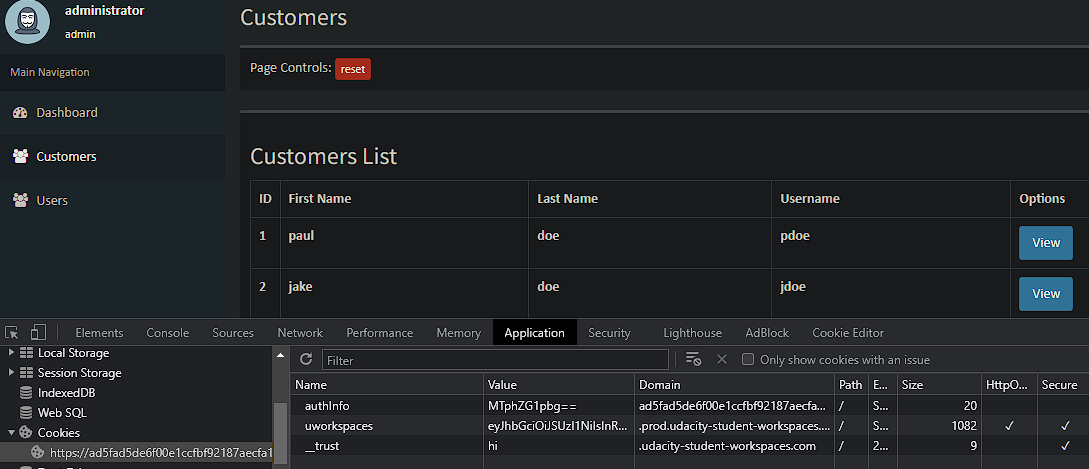
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

A broken access vulnerability was exploited, allowing the attacker to modify the cookies of Guest users to match those of the admin, there by gaining unauthorized access to the Users page.

**Vulnerability Walk-thru**:

1. First go to the Customers page while you are opening the Developer options-> application -> storage -> cookies 
2. Modify the Authinfo value with “MTphZG1pbg==” then reload the page 
3. You will be able to view the Customers List

**Recommendations:**

it is recommended to implement a strong access control mechanism that ensures proper user authorization checks and restricts access based on roles and permissions.

<https://owasp.org/www-project-top-ten/2017/A5_2017-Broken_Access_Control.html>

**VWAYYMMDD## – Sensitive Data Exposure- High**

**Vulnerability Exploited:** **Sensitive Data Exposure**

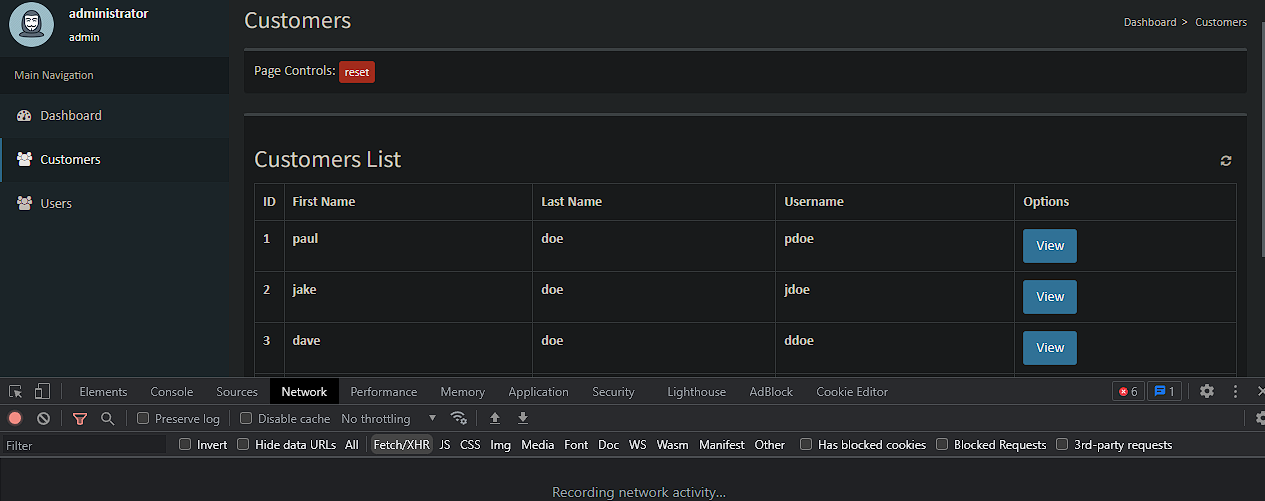
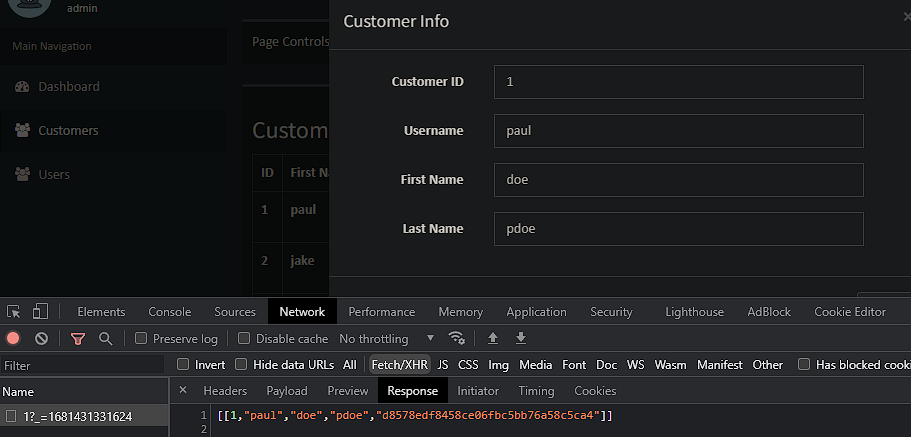
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

A sensitive data exposure vulnerability was identified, where the attacker was able to access hash values for customer data by inspecting network requests in the browser's developer options.

**Vulnerability Walk-thru**:

1. Go to the customers page while you are opening the Developer options -> network -> XHR
2. View any user from the list then investigate the request from the Developer options by checking the response 
3. You will be able to see the hash of the user.

**Recommendations:**

it is recommended to implement proper access controls and authentication mechanisms to ensure that sensitive data such as user passwords cannot be accessed by unauthorized users

<https://owasp.org/www-project-proactive-controls/v3/en/c8-protect-data-everywhere>

**VWAYYMMDD## – Security Misconfiguration - High**

**Vulnerability Exploited:** **Security Misconfiguration**

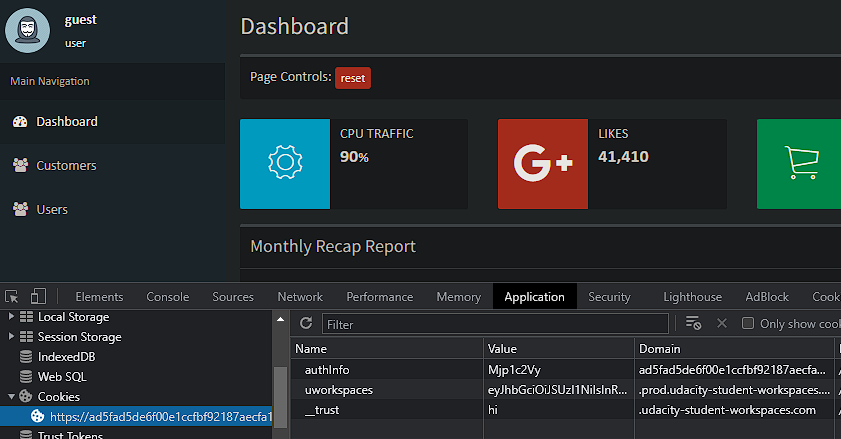
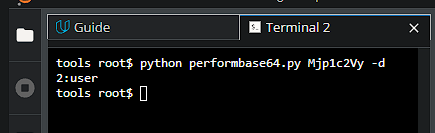
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

The Developer used Base64 to encode the user’s cookie

**Vulnerability Walk-thru**:

1. Open the developer options -> application ->storage ->Cookies
2. Copy the value of the Authinfo then check it with “performbase64.py” script by adding the command to the terminal in the workspace:   
   python performbase64.py Mjp1c2Vy -d
3. You will notice that the developer used base64 to encode the cookies value

**Recommendations:**

it is recommended to implement secure coding practices such as using appropriate encryption and encoding techniques for sensitive data such as cookies.

<https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration/>

**VWAYYMMDD## – Security Misconfiguration - High**

**Vulnerability Exploited:** **Security Misconfiguration**

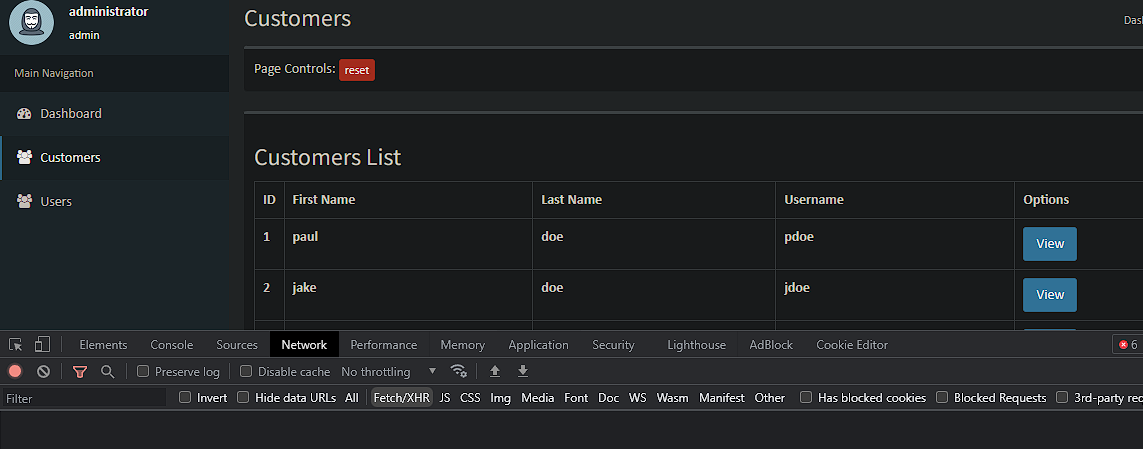
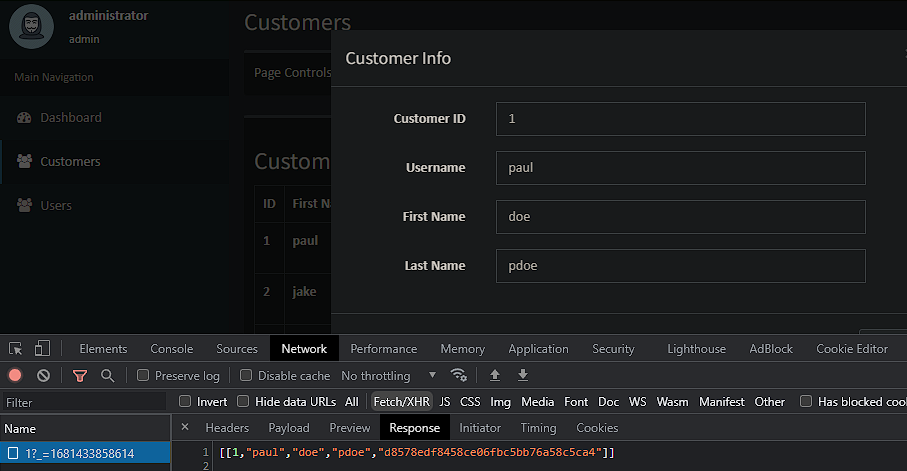
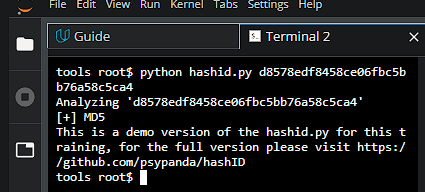
**Severity: High**

**System:** VWA Web Application

**Vulnerability Explanation**:

The Developer Used md5 to Hash the user’s Password.

**Vulnerability Walk-thru**:

1. Login as admin and go to the customers page while opening the developer options -> network ->xhr
2. View any user from the “view” Button then check the response and copy the hash value 
3. On the workspace use the “hashid.py” script to check the hash type 
4. It’s md5 hash which not recommended to use for hashing passwords

**Recommendations:**

It is recommended to use a stronger cryptographic hash function such as bcrypt.

<https://owasp.org/www-project-top-ten/2017/A6_2017-Security_Misconfiguration.html>

**VWAYYMMDD## – SQLI - High**

**Vulnerability Exploited:** **SQLI**

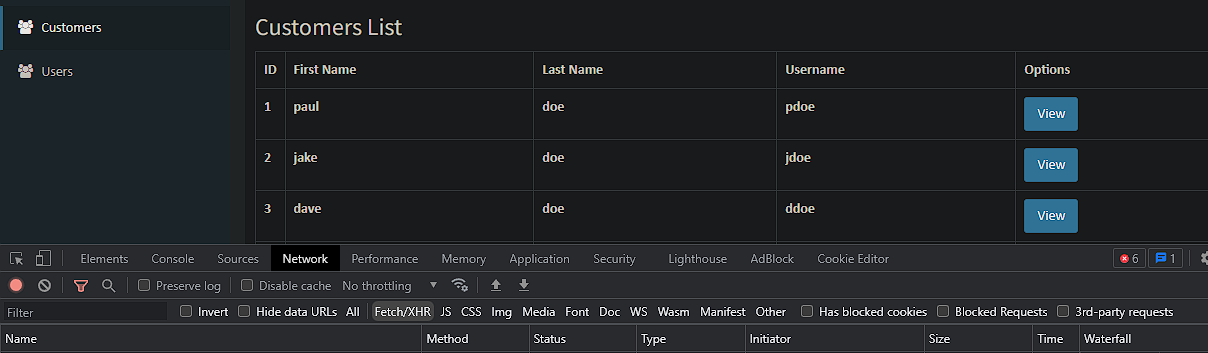
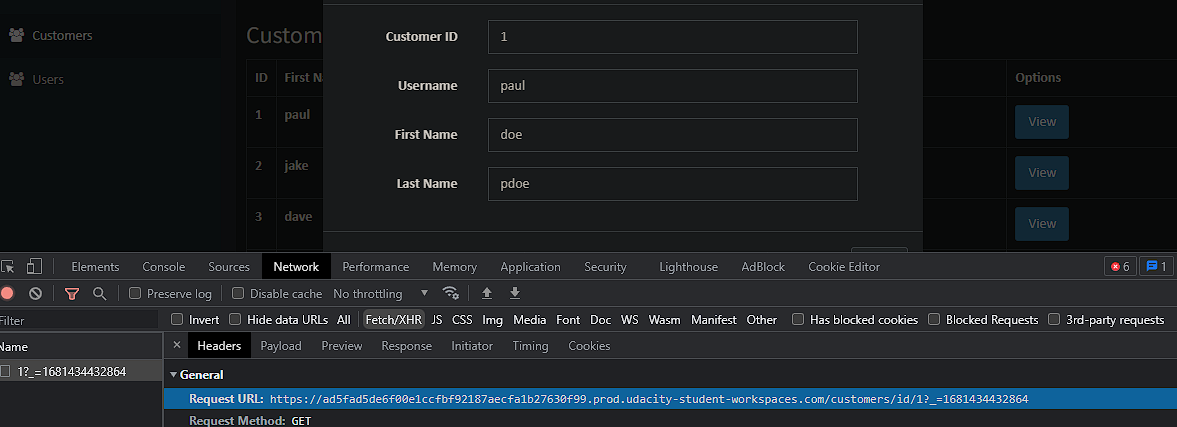
**Severity: High**

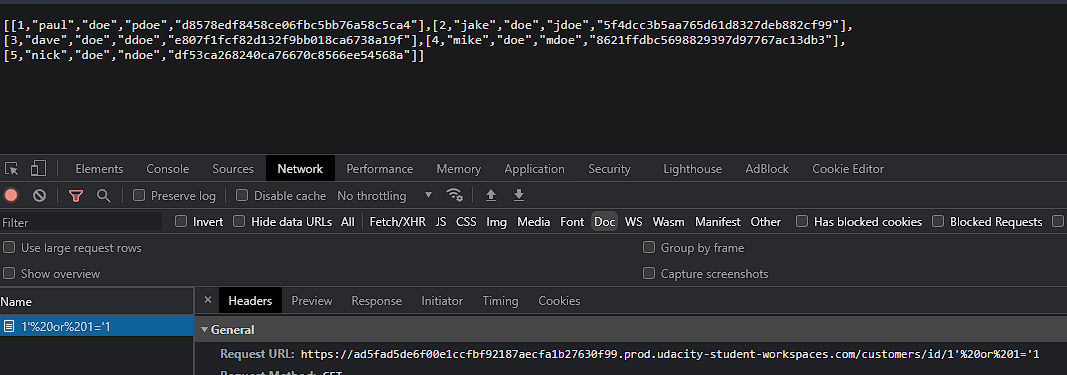
**System:** VWA Web Application

**Vulnerability Explanation**:

The web application's customer page was found to be vulnerable to SQL injection attacks, allowing the attacker to manipulate the SQL queries and retrieve unauthorized data from the database.

**Vulnerability Walk-thru**:

1. Login as administrator then go to customer page while you opening the Developer options -> Network -> XHR 
2. View one of the customers by clicking on “View” button, then from the Headers copy The Request URL
3. Open this link in a new tab From this link: [https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1A screenshot of a computer

   Description automatically generated](https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1)
4. Now in the URL add this payload: https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1’ or 1=’1
5. It will retrieve the whole customer’s list data

**Recommendations:**

It is recommended to implement proper input validation and sanitization measures such as parameterized queries.

https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Pre

vention\_Cheat\_Sheet.html

**VWAYYMMDD## – SQLI - High**

**Vulnerability Exploited:** **SQLI**

**Severity: High**

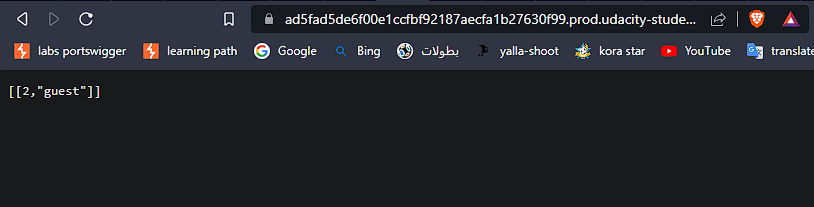
**System:** VWA Web Application

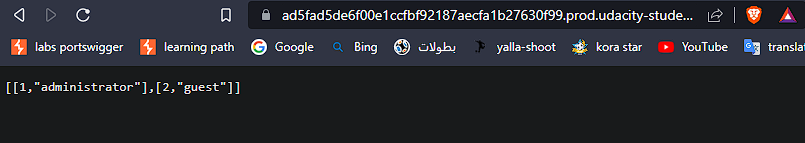
**Vulnerability Explanation**:

This endpoint in The web application /profile/userlist page was found to be vulnerable to SQL injection attacks, allowing the attacker to manipulate the SQL queries and retrieve unauthorized data from the database.

**Vulnerability Walk-thru**:

1. Login to application then go to this endpoint:  
   <https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/profile/userlist/1>



1. In the URL add the following payload:  
   <https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/profile/userlist/1>’ or 1=’1
2. It will retrieve all the Users from the userlist

**Recommendations:**

It is recommended to implement proper input validation and sanitization measures such as parameterized queries.

https://cheatsheetseries.owasp.org/cheatsheets/SQL\_Injection\_Pre

vention\_Cheat\_Sheet.html