VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWAYYMMDD
Author: [Ahmad Mahmoud]
Date: [13-4-2023]

VWA Security Report

VWA Security Report

VWAYYMMDD## - Broken Auth - High

Vulnerability Exploited: Broken Auth using Bruteforce

Severity: High

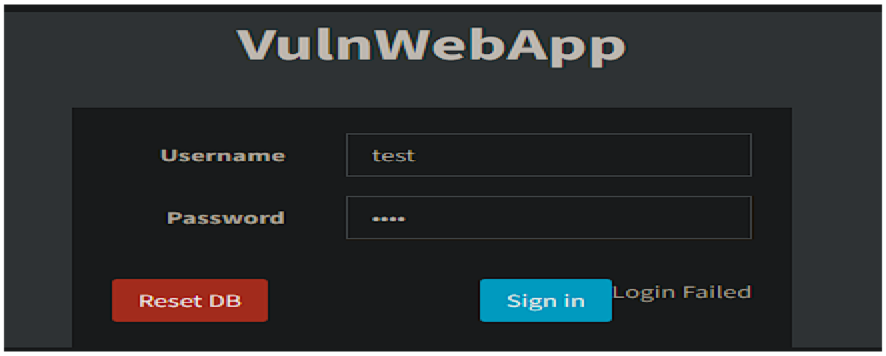
System: VWA Web Application

Vulnerability Explanation:

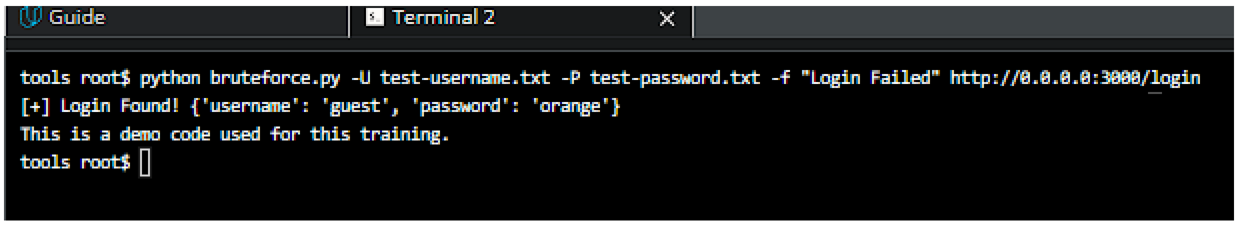
An unauthorized user used a brute force attack against the login form of a web application and was able to access the application.

Vulnerability Walk-thru:

1) First go to the login page and enter invalid credentials.

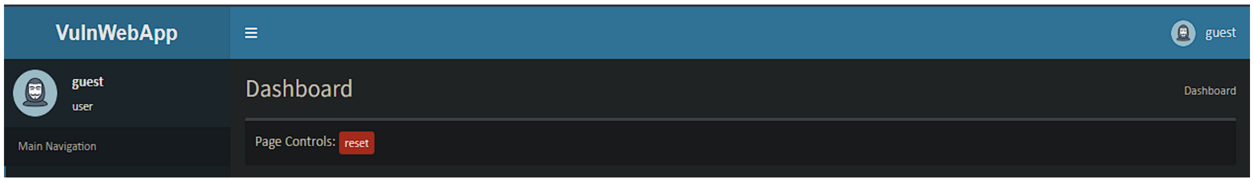


2) Go to the workspace terminal and use the "bruteforce.py" script and use this command: python bruteforce.py -U test-username.txt -P test-password.txt -f "Login Failed" http://0.0.0.0:3000/login



```
tools root$ python bruteforce.py -U test-username.txt -P test-password.txt -f "Login Failed" http://0.0.0.0:3000/login
[+] Login Found! {'username': 'guest', 'password': 'orange'}
This is a demo code used for this training.
tools root$ 
```

3) Enter the credentials you got from the script



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.

VWA Security Report

https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

VWAYYMMDD## - XSS - High

Vulnerability Exploited: XSS

Severity: High

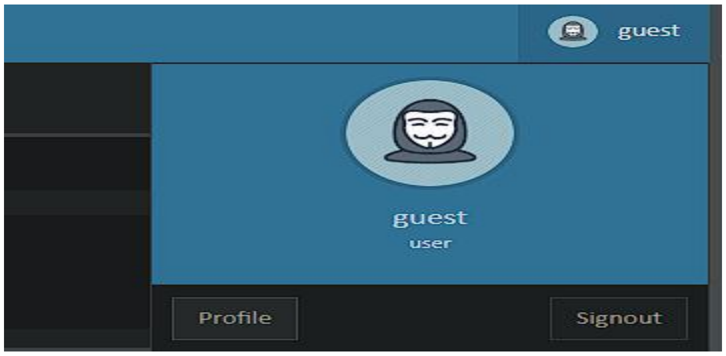
System: VWA Web Application

Vulnerability Explanation:

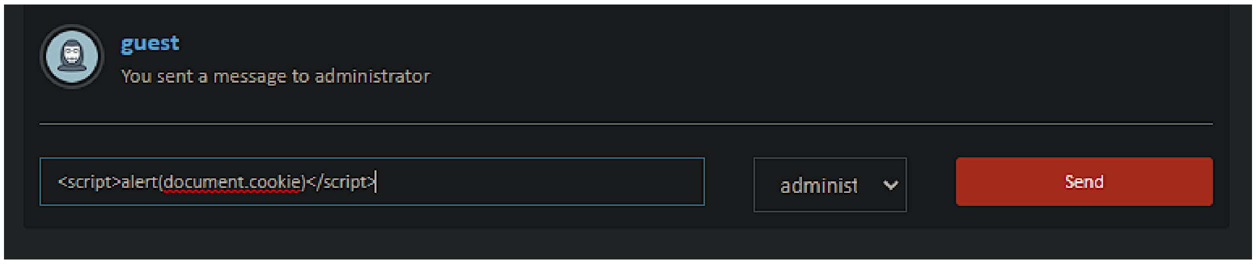
In the Profile Area we have input field allow you to send message to the Administrator, this input field seems to not be sanitizing for "script" tag within the messages That then could contain java-script code and is executable on the client side.

Vulnerability Walk-thru:

1) First go to the profile section



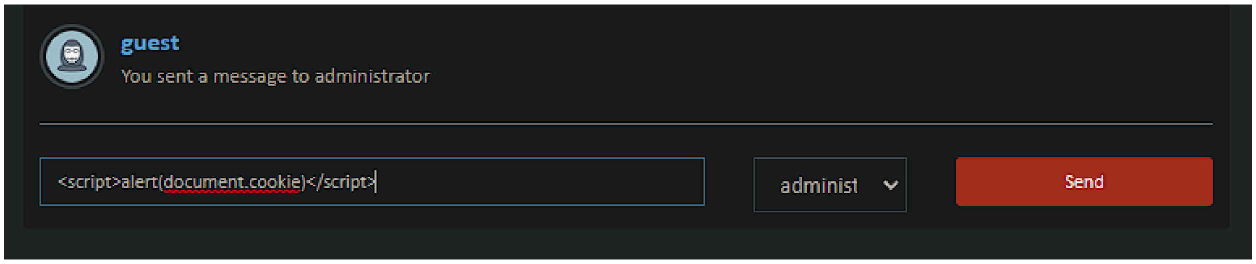
2) In the input field add this payload: `<script>alert(document.cookie)</script>`



You sent a message to administrator

<script>alert(document.cookie)</script>

3) When you reload the page it will response with the cookies of the user


guest
You sent a message to administrator

```
<script>alert(document.cookie)</script>
```

Recommendations:

We should use a standard library that does field sanitizing for XSS.
https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html

---

VWAYYMMDD## – Insecure Deserialization – High

Vulnerability Exploited: **Insecure Deserialization**

Severity: **High**

System: VWA Web Application

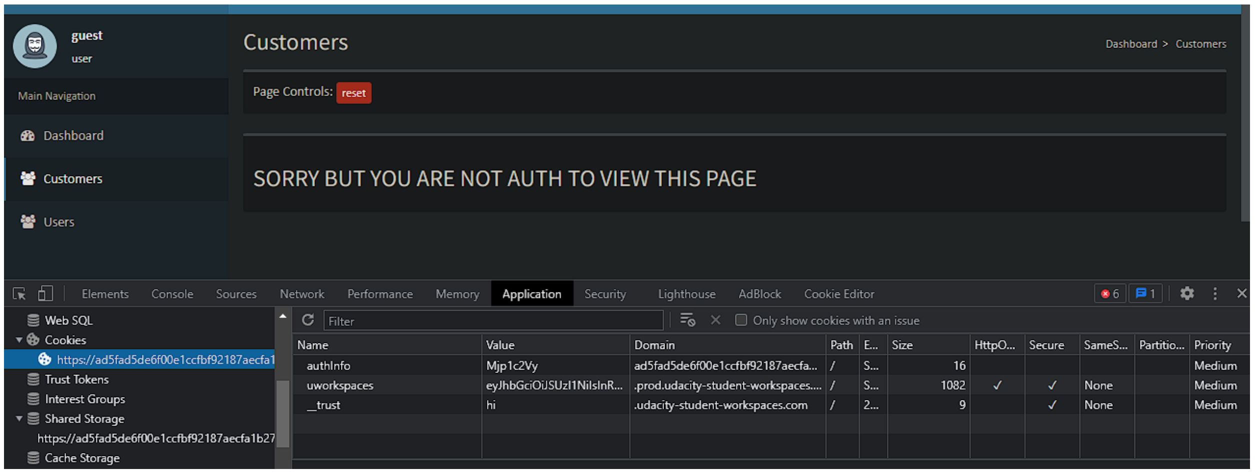
Vulnerability Explanation:

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:

VWA Security Report

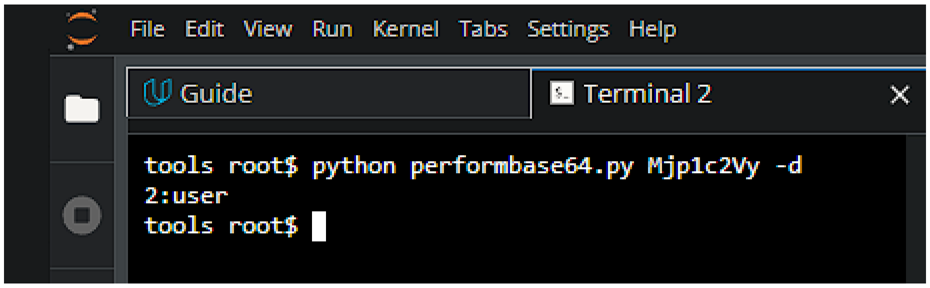
1) First go to the customer's page while you are opening the Developers options -> application -> storage-> cookies



SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE

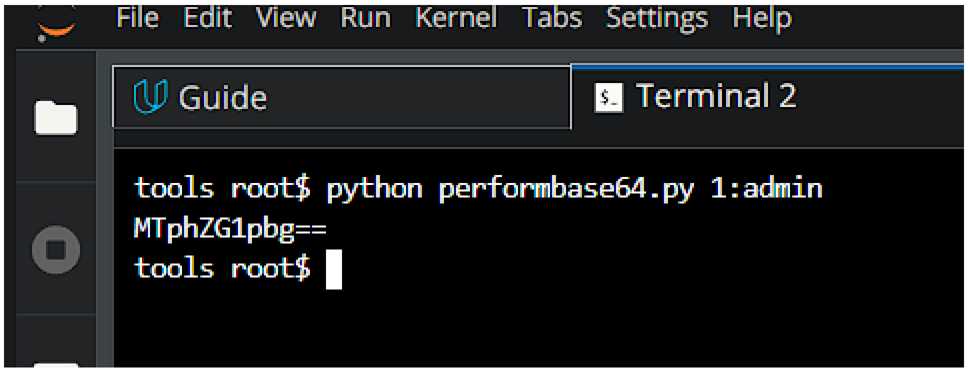
Name	Value	Domain	Path	E...	Size	HttpO...	Secure	SameS...	Partitio...	Priority
authInfo	Mjp1c2Vy	ad5fad5de6f00e1ccfbf92187aecfa...	/	S...	16					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....	/	S...	1082	✓	✓	None		Medium
__trust	hi	.udacity-student-workspaces.com	/	2...	9		✓	None		Medium

2) Let's check the Authinfo value with "performbase64.py" from the workspace.



```
tools root$ python performbase64.py Mjp1c2Vy -d
2:user
tools root$
```

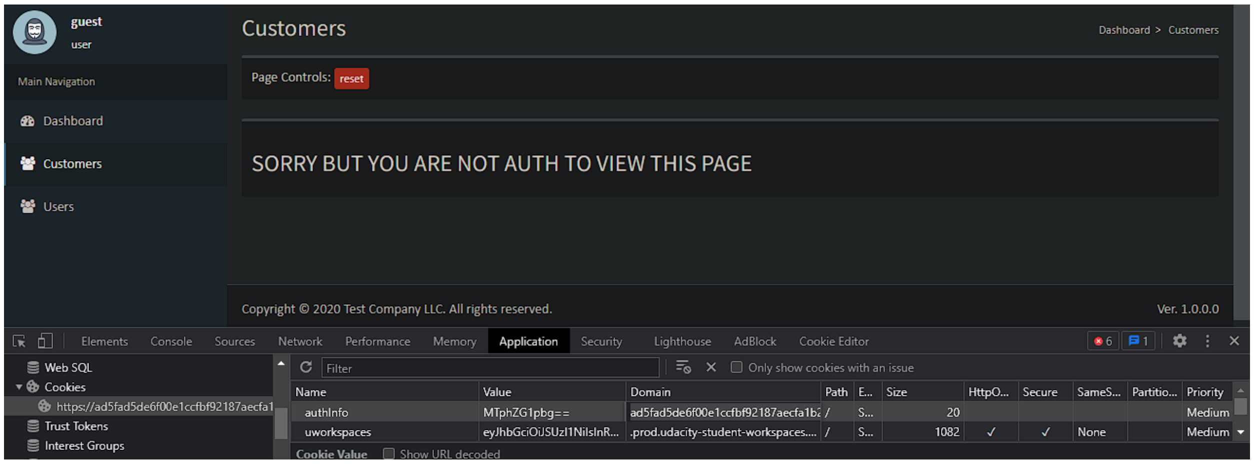
3) Now convert 1:admin to base64 with "performbase64.py"



```
tools root$ python performbase64.py 1:admin
MTphZG1pbg==
tools root$
```

VWA Security Report

4) Then modify the Authinfo value with MTphZG1pbg== and reload the page of customers



Customers

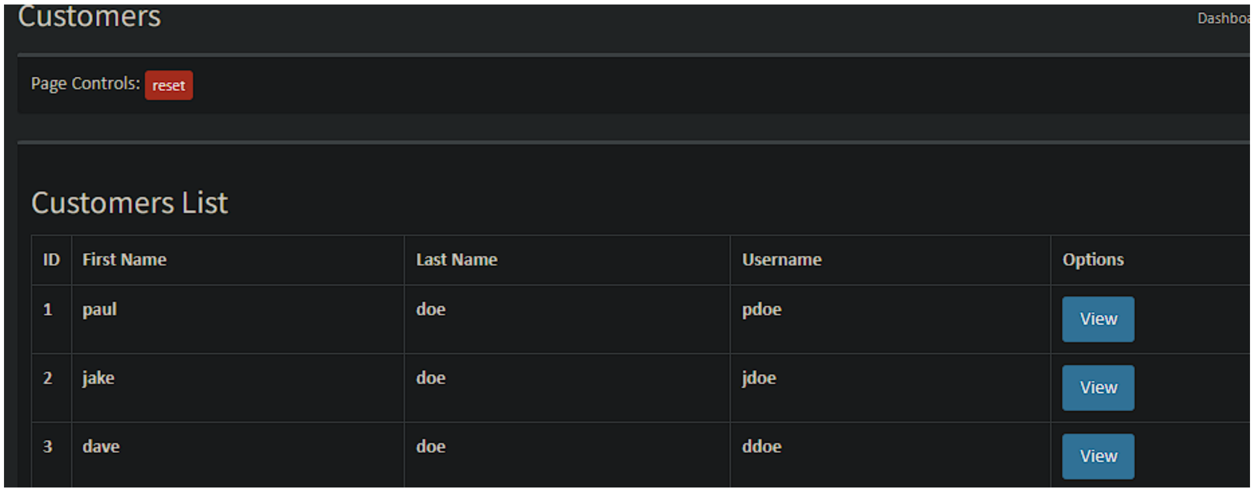
SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE

Copyright © 2020 Test Company LLC. All rights reserved.

Ver. 1.0.0.0

Name	Value	Domain	Path	E...	Size	HttpO...	Secure	SameS...	Partitio...	Priority
authInfo	MTphZG1pbg==	ad5fad5de6f00e1ccfbf92187aecfa1b2	/	S...	20					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....	/	S...	1082	✓	✓	None		Medium

5) You will able to see the Customers list



Customers

Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View

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

VWA Security Report

VWAYYMMDD## –Broken Access – High

Vulnerability Exploited: Broken Access to Users

Severity: High

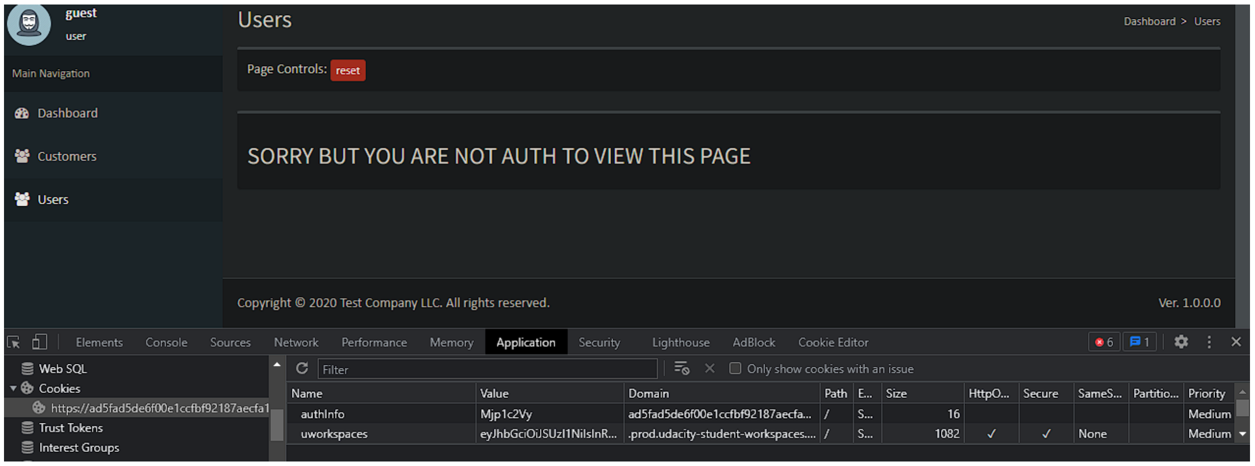
System: VWA Web Application

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


guest
user
Users
Dashboard > Users
Page Controls: reset
Main Navigation
Dashboard
Customers
Users
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Web SQL
Cookies
Trust Tokens
Interest Groups
Filter
Only show cookies with an issue

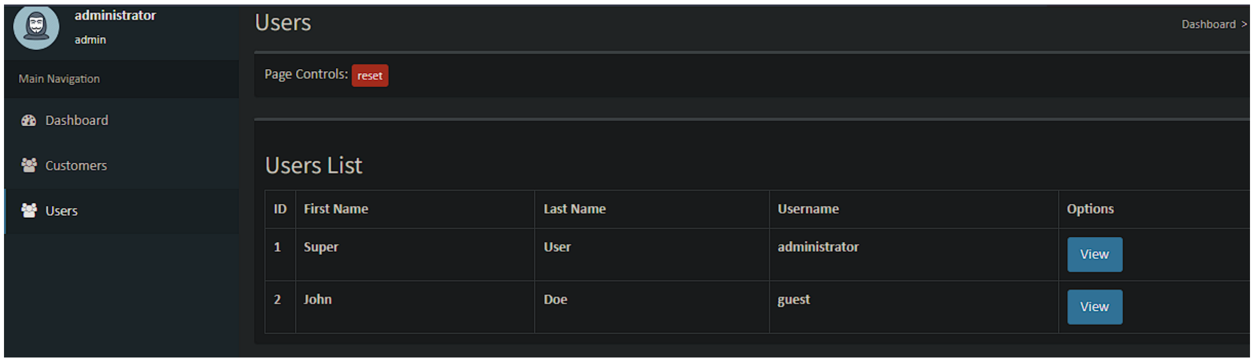
SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE

Copyright © 2020 Test Company LLC. All rights reserved. Ver. 1.0.0.0

https://ad5fad5de6f00e1ccfbf92187aecfa1

Name	Value	Domain	Path	E...	Size	HttpO...	Secure	SameS...	Partitio...	Priority
authInfo	Mjp1c2Vy	ad5fad5de6f00e1ccfbf92187aecfa...	/	S...	16					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....	/	S...	1082	✓	✓	None		Medium

2) Modify the Authinfo value with "MTphZG1pbg==" then reload the page



Users List

ID	First Name	Last Name	Username	Options
1	Super	User	administrator	View
2	John	Doe	guest	View

VWA Security Report

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**

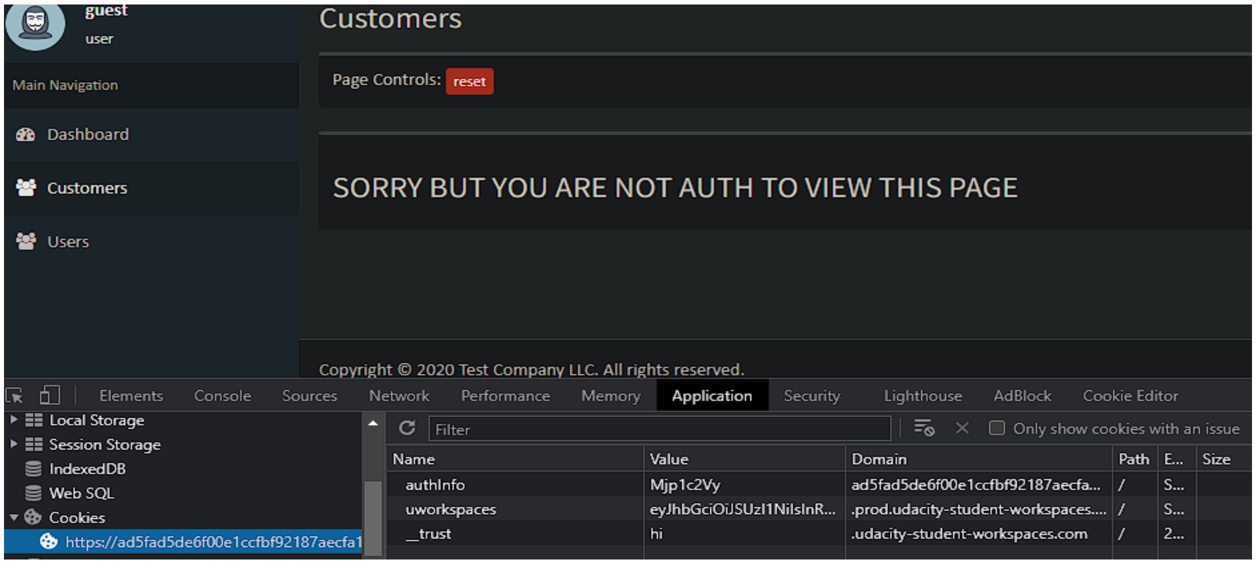
System: VWA Web Application

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

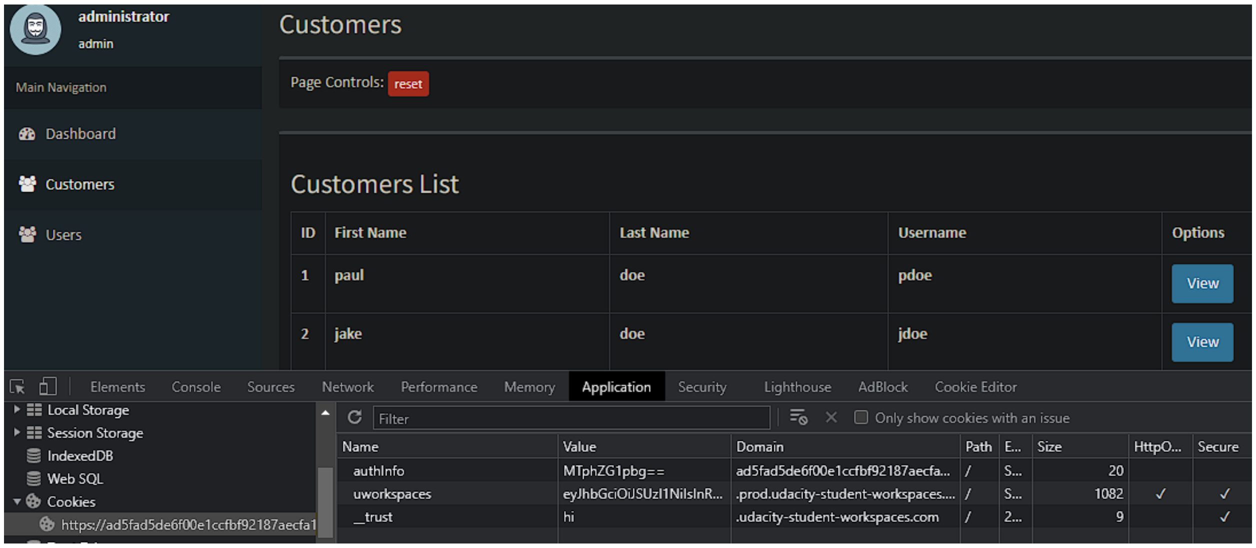


SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE

Copyright © 2020 Test Company LLC. All rights reserved.

Name	Value	Domain	Path	E...	Size
authInfo	Mjp1c2Vy	ad5fad5de6f00e1ccfbf92187aecfa...	/	S...	
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....	/	S...	
__trust	hi	.udacity-student-workspaces.com	/	2...	

2) Modify the Authinfo value with "MTphZG1pbg==" then reload the page



Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View

Name	Value	Domain	Path	E...	Size	HttpO...	Secure
authInfo	MTphZG1pbg==	ad5fad5de6f00e1ccfbf92187aecfa...	/	S...	20		
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....	/	S...	1082	✓	✓
__trust	hi	.udacity-student-workspaces.com	/	2...	9		✓

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

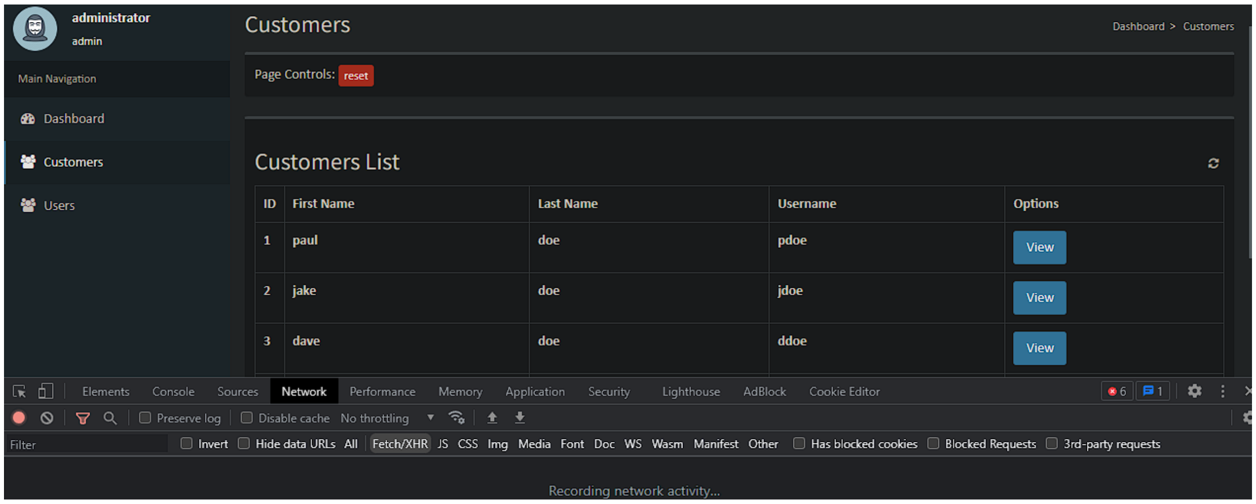
Vulnerability Explanation:

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.

VWA Security Report

Vulnerability Walk-thru:

1) Go to the customers page while you are opening the Developer options -> network -> XHR

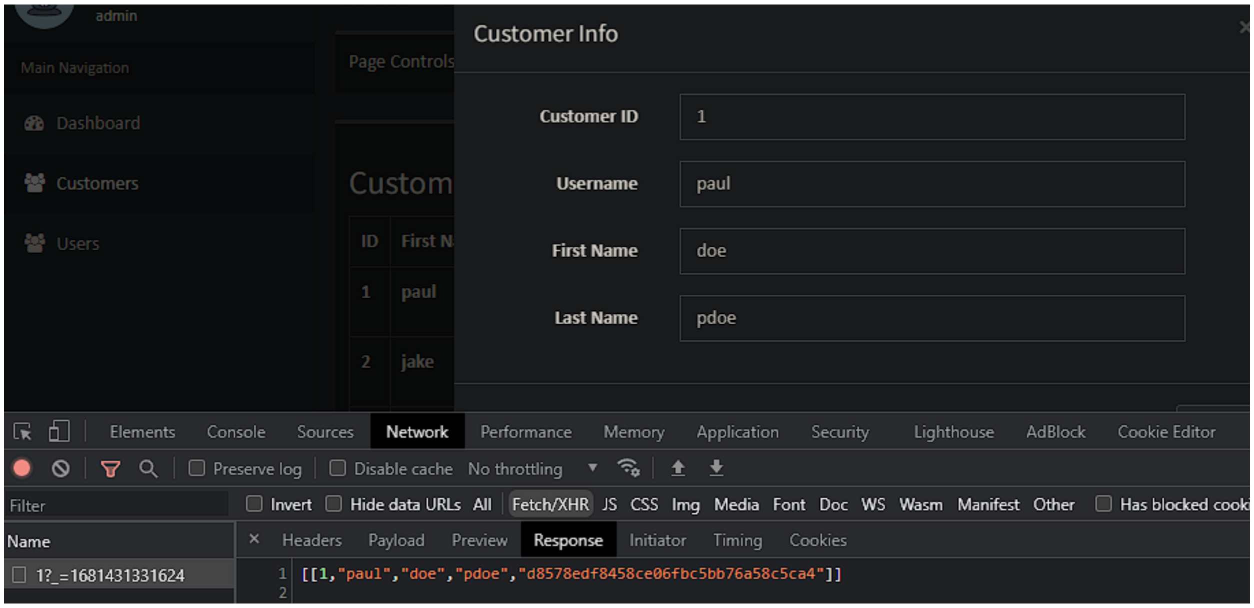

administrator
admin
Main Navigation
Dashboard
Customers
Users
Dashboard > Customers
Page Controls: reset
View
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests
Recording network activity...

Customers

Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View

2) View any user from the list then investigate the request from the Developer options by checking the response


admin
Main Navigation
Dashboard
Customers
Users
Page Controls
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cook
Name
Headers Payload Preview Response Initiator Timing Cookies

Customer Info

Customer ID	1
Username	paul
First Name	doe
Last Name	pdoe

1?_=1681431331624

```
[[1,"paul","doe","pdoe","d8578edf8458ce06fbc5bb76a58c5ca4"]]
```

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

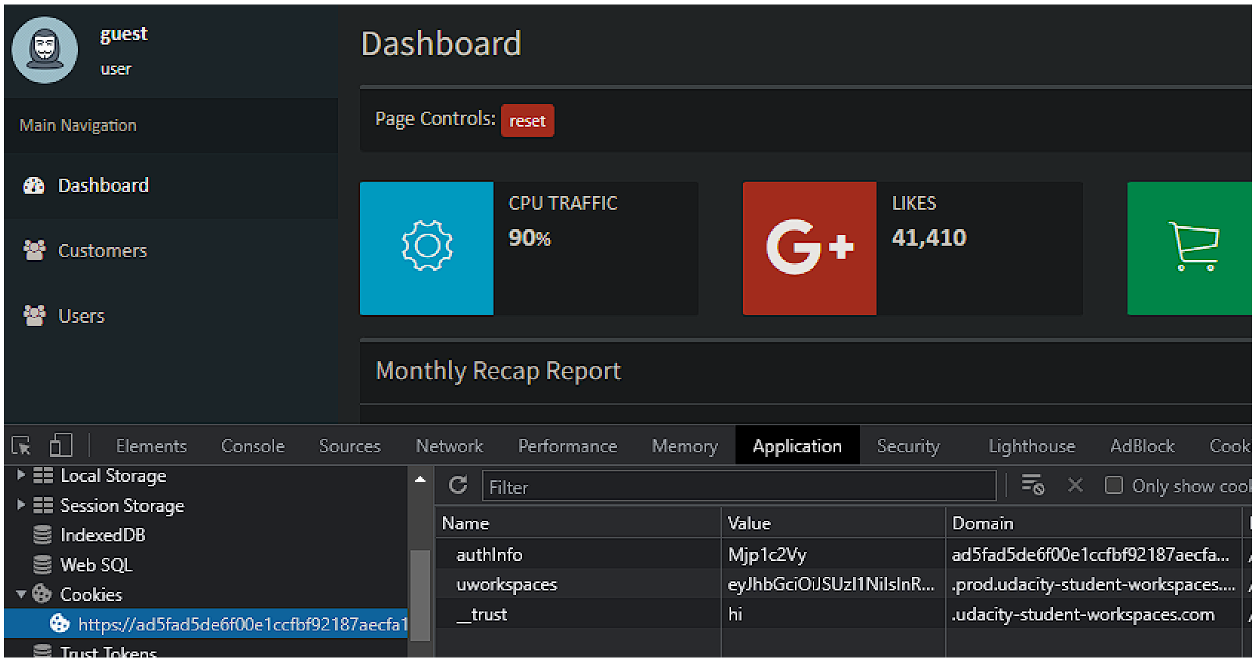
System: VWA Web Application

Vulnerability Explanation:

The Developer used Base64 to encode the user's cookie

Vulnerability Walk-thru:

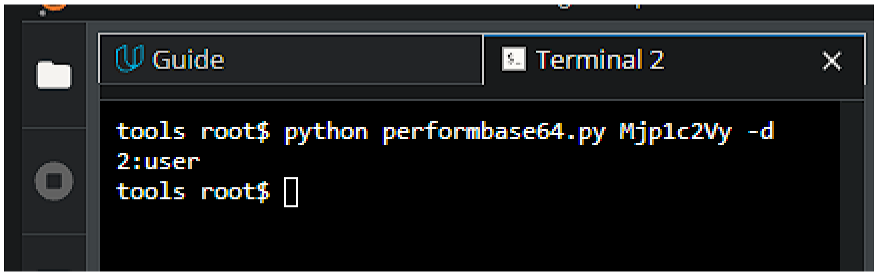
1) Open the developer options -> application ->storage ->Cookies



Name	Value	Domain
authInfo	Mjp1c2Vy	ad5fad5de6f00e1ccfbf92187aecfa...
uworkspaces	eyJhbGciOiJSUzI1NiIsInR...	.prod.udacity-student-workspaces....
__trust	hi	.udacity-student-workspaces.com

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



```
tools root$ python performbase64.py Mjp1c2Vy -d
2:user
tools root$ 
```

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/

VWA Security Report

VWAYYMMDD## – Security Misconfiguration – High

Vulnerability Exploited: Security Misconfiguration

Severity: High

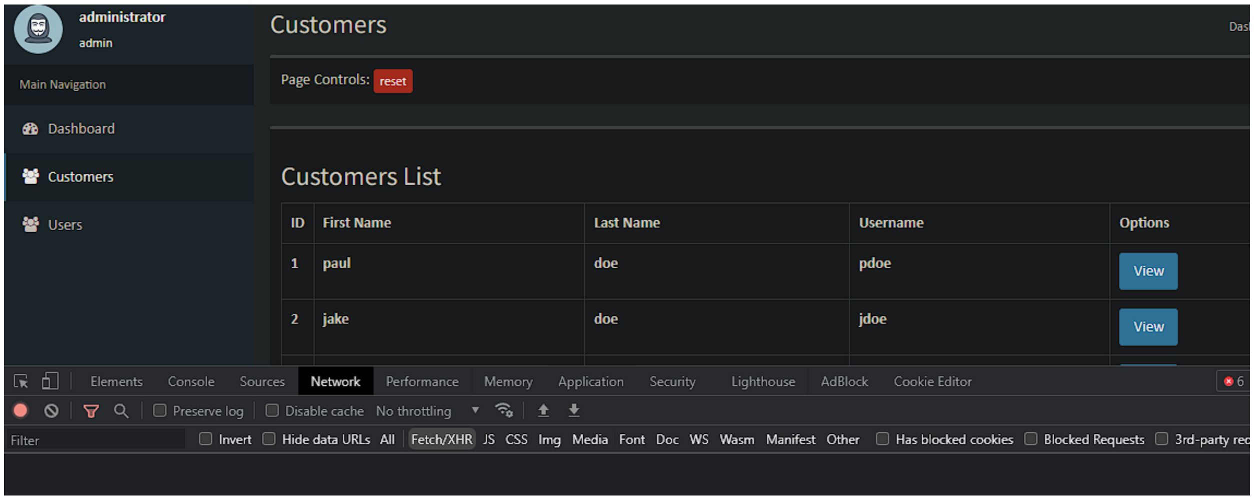
System: VWA Web Application

Vulnerability Explanation:

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

Vulnerability Walk-thru:

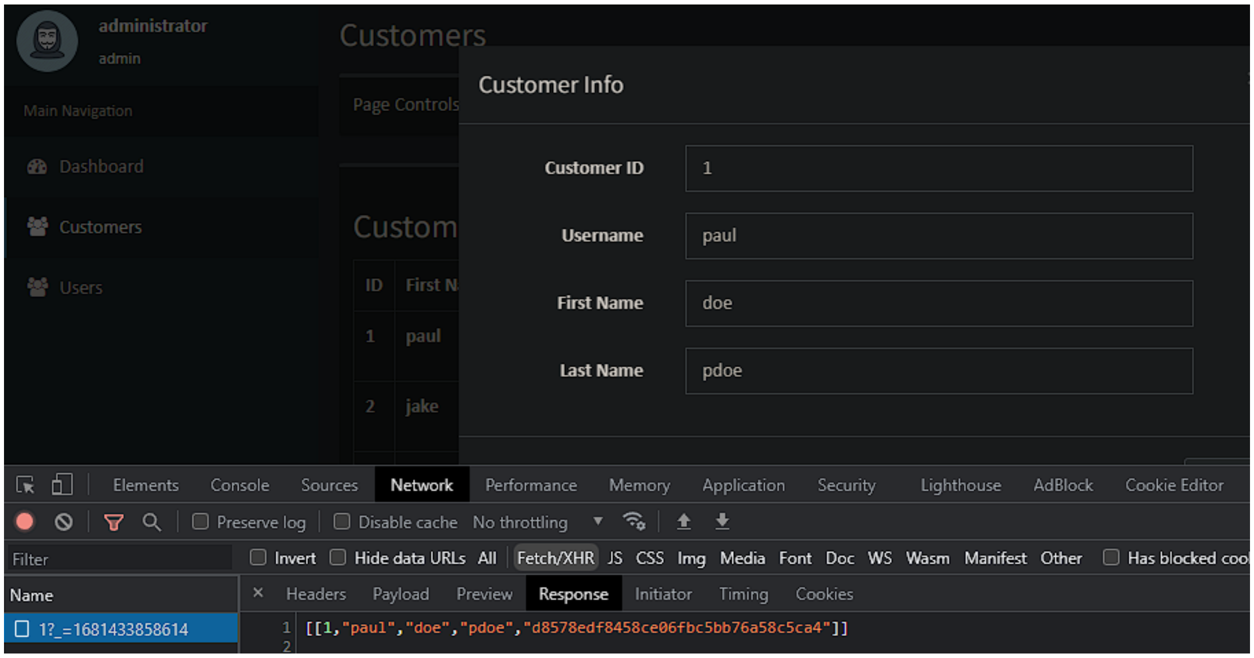
1) Login as admin and go to the customers page while opening the developer options -> network ->xhr


administrator
admin
Main Navigation
Dashboard
Customers
Users
Customers
Page Controls: reset
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party re

Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View

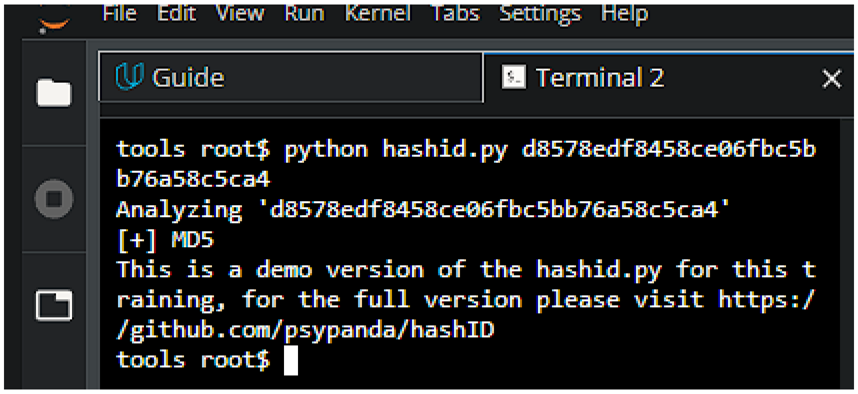
2) View any user from the "view" Button then check the response and copy the hash value



Customer ID	1
Username	paul
First Name	doe
Last Name	pdoe

```
1?_=1681433858614
[[1,"paul","doe","pdoe","d8578edf8458ce06fbc5bb76a58c5ca4"]]
```

3) On the workspace use the "hashid.py" script to check the hash type



```
tools root$ python hashid.py d8578edf8458ce06fbc5b
b76a58c5ca4
Analyzing 'd8578edf8458ce06fbc5bb76a58c5ca4'
[+] MD5
This is a demo version of the hashid.py for this t
raining, for the full version please visit https:/
/github.com/psypanda/hashID
tools root$
```

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

VWA Security Report

VWAYYMMDD## – SQLI – High

Vulnerability Exploited: SQLI

Severity: High

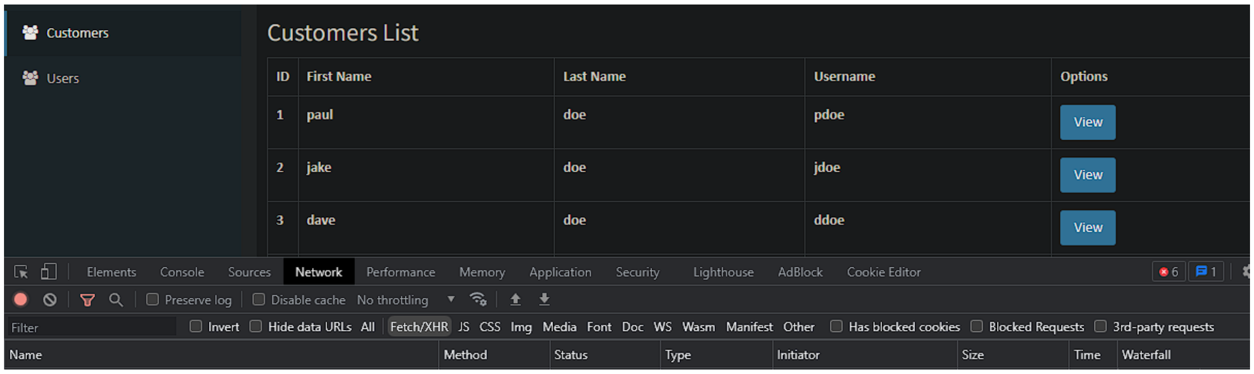
System: VWA Web Application

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

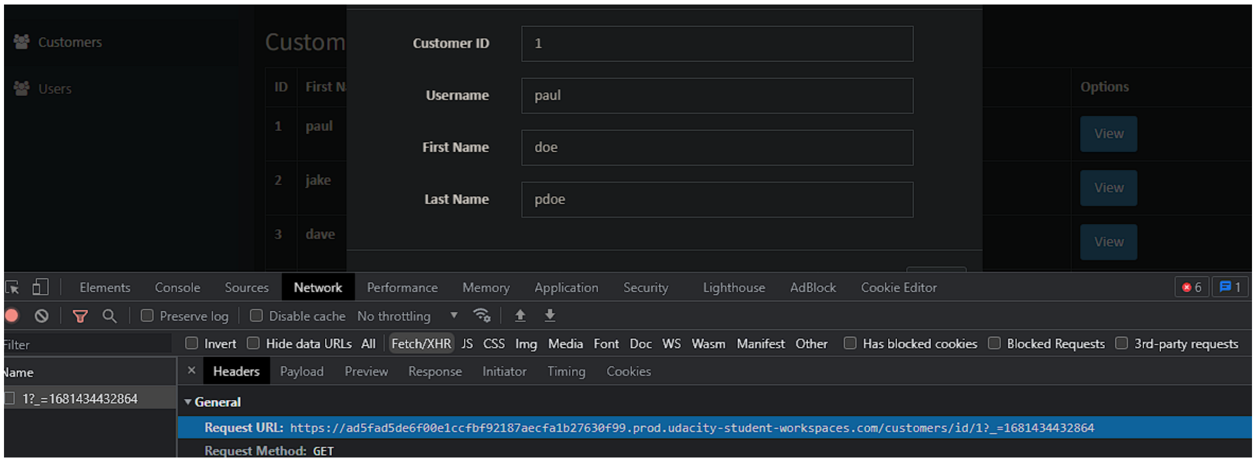


Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View

Customers
Users
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests
Name Method Status Type Initiator Size Time Waterfall

2) View one of the customers by clicking on "View" button, then from the Headers copy The Request URL



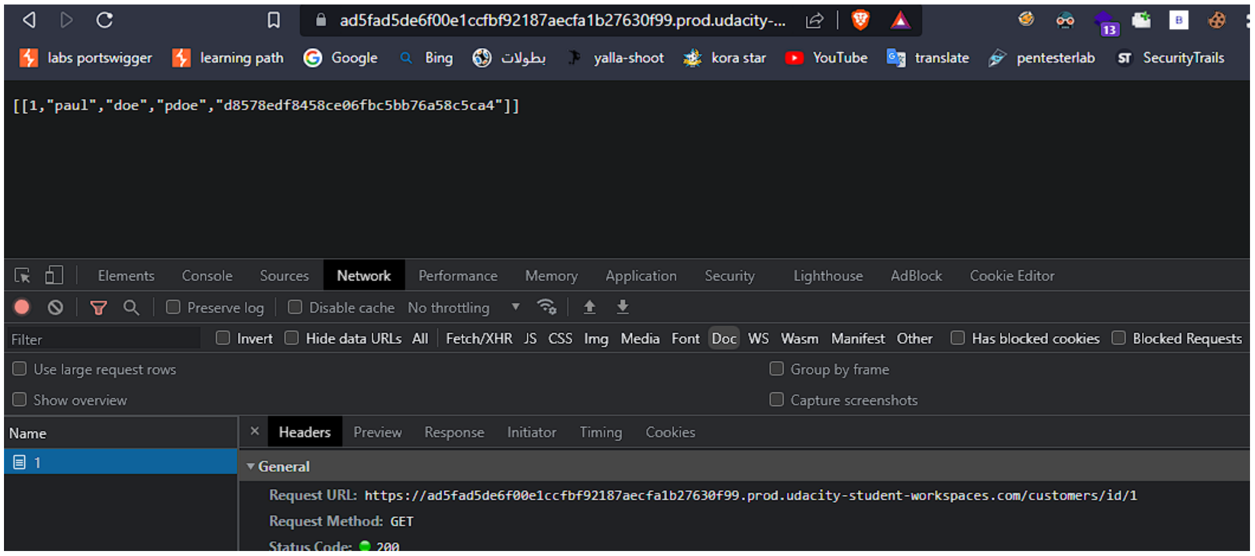
Customer ID	1
Username	paul
First Name	doe
Last Name	pdoe

1?_=1681434432864

General
Request URL: https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1?_=1681434432864
Request Method: GET

Customers
Users
Elements Console Sources Network Performance Memory Application Security Lighthouse AdBlock Cookie Editor
Preserve log Disable cache No throttling
Filter Invert Hide data URLs All Fetch/XHR JS CSS Img Media Font Doc WS Wasm Manifest Other Has blocked cookies Blocked Requests 3rd-party requests
Name
Headers Payload Preview Response Initiator Timing Cookies

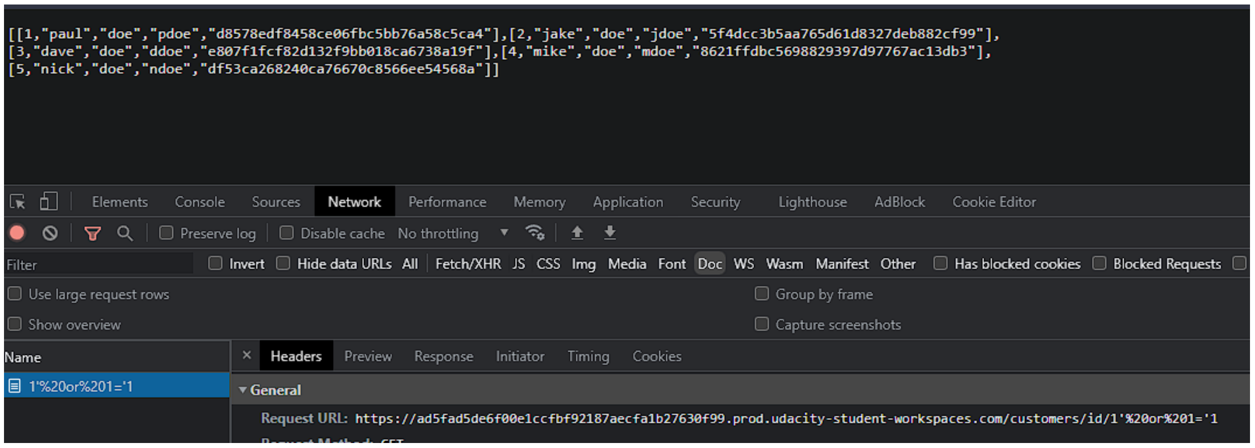
3) Open this link in a new tab From this link: https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1



```
[[1,"paul","doe","pdoe","d8578edf8458ce06fbc5bb76a58c5ca4"]]
```

Request URL: https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1
Request Method: GET
Status Code: 200

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



```
[[1,"paul","doe","pdoe","d8578edf8458ce06fbc5bb76a58c5ca4"],[2,"jake","doe","jdoe","5f4dcc3b5aa765d61d8327deb882cf99"],
[3,"dave","doe","ddoe","e807f1fcf82d132f9bb018ca6738a19f"],[4,"mike","doe","mdoe","8621ffdbc5698829397d97767ac13db3"],
[5,"nick","doe","ndoe","df53ca268240ca76670c8566ee54568a"]]
```

Name: 1'%20or%201='1

Request URL: https://ad5fad5de6f00e1ccfbf92187aecfa1b27630f99.prod.udacity-student-workspaces.com/customers/id/1'%20or%201='1

Recommendations:

It is recommended to implement proper input validation and sanitization measures such as parameterized queries.
https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html

VWA Security Report

VWAYYMMDD## – SQLI – High

Vulnerability Exploited: SQLI

Severity: High

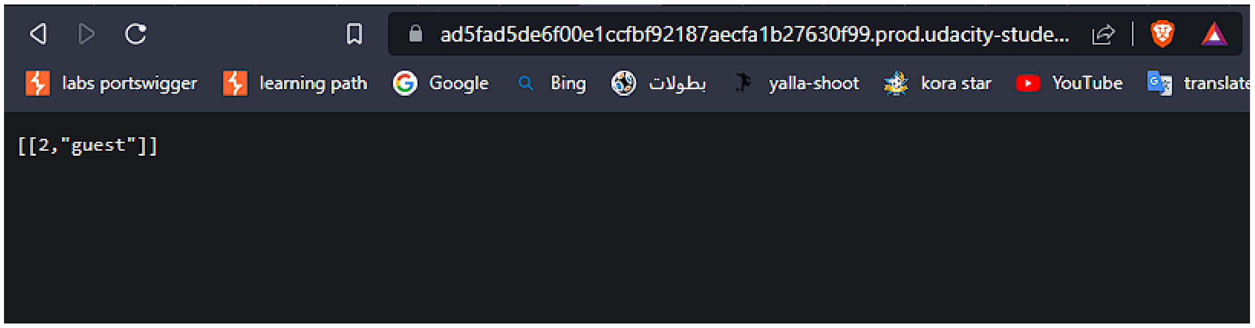
System: VWA Web Application

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.

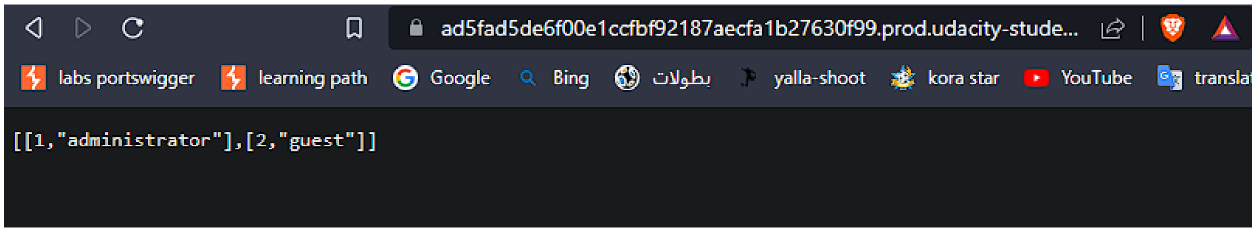
Vulnerability Walk-thru:

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



```
[[2,"guest"]]
```

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



```
[[1,"administrator"],[2,"guest"]]
```

Recommendations:

It is recommended to implement proper input validation and sanitization measures such as parameterized queries. https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html