

VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWAYMMDD
Author: [Ki Man Ho Chico]
Date: [1/12/21]

VWA Security Report

VWAYYMMDD## - A6:2017 - Security Misconfiguration - HIGH	3
VWAYYMMDD## - A3:2017 - Sensitive Data Exposure - MEDIUM	5
VWAYYMMDD## - A1:2017 - Injection - MEDIUM	13
VWAYYMMDD## - A1:2017 - Injection - MEDIUM	17
VWAYYMMDD## - A7:2017 - Cross-Site Scripting (XSS) - MEDIUM	21
VWAYYMMDD## - A2:2017 - Broken Authentication - MEDIUM	25
VWAYYMMDD## - A5:2017 - Broken Access Control - MEDIUM	28
VWAYYMMDD## - A5:2017 - Broken Access Control - MEDIUM	33
VWAYYMMDD## - A5:2017 - Broken Access Control - LOW	36
VWAYYMMDD## - A8:2017 - Insecure Deserialization - LOW	39

VWA Security Report

VWAYMMDD## - A6:2017 - Security

Misconfiguration - HIGH

Vulnerability Exploited: A6 - Security Misconfiguration

Severity: [High]

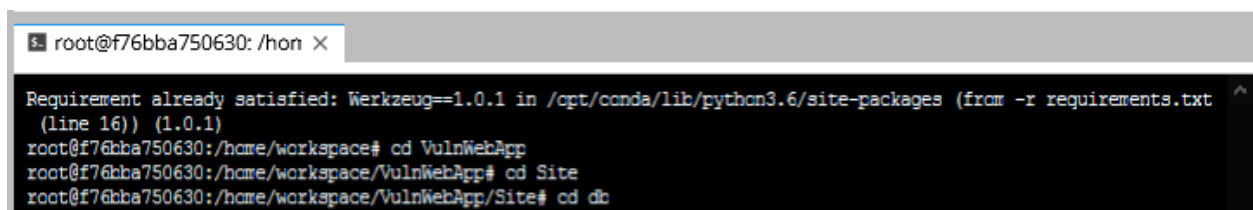
System: VWA Web Application

Vulnerability Explanation:

Security Misconfiguration is basically distinct as deteriorating to implement all the security controls for a server or web application. It is commonly a result of insecure default configurations, incomplete or ad hoc configurations, open cloud storage, misconfigured HTTP headers, and verbose error messages containing sensitive information.

Vulnerability Walk-thru:

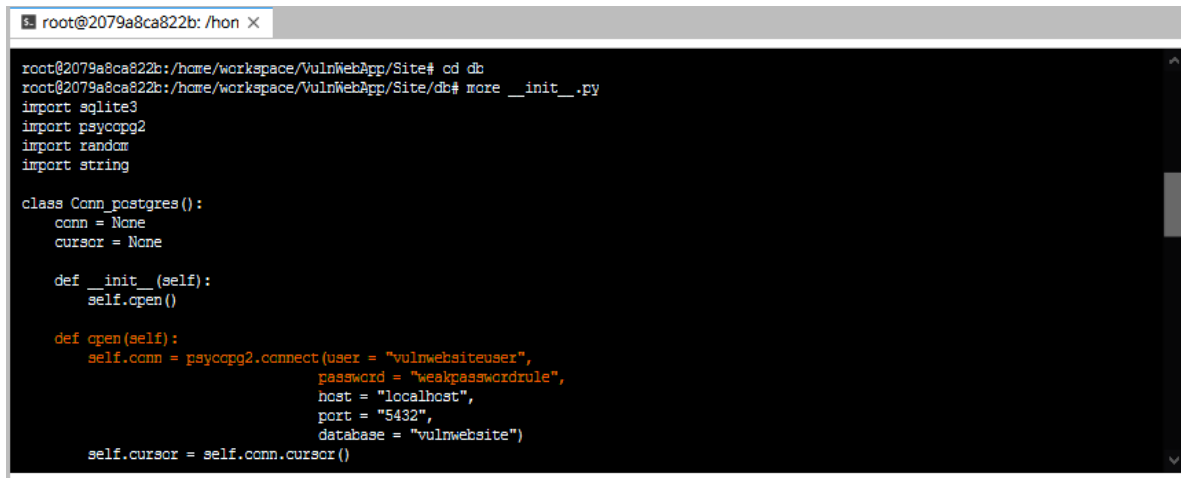
By scanning VulnWebApp/Site/db/more `__init__.py` file one security misconfiguration found regarding password issue.



```
root@f76bba750630: /home/x
Requirement already satisfied: Werkzeug==1.0.1 in /opt/conda/lib/python3.6/site-packages (from -r requirements.txt
(line 16)) (1.0.1)
root@f76bba750630:/home/workspace# cd VulnWebApp
root@f76bba750630:/home/workspace/VulnWebApp# cd Site
root@f76bba750630:/home/workspace/VulnWebApp/Site# cd db
```

VWA Security Report

Issue: Possible hardcoded password: 'weakpasswordrule'

A terminal window with a dark background and light-colored text. The title bar shows 'root@2079a8ca822b: /hon x'. The terminal content shows a user navigating to a directory and viewing a Python file. The Python code defines a class 'Conn_postgres' with methods for database connection. A password 'weakpasswordrule' is hardcoded in the 'open' method.

```
root@2079a8ca822b:/home/workspace/VulnWebApp/Site# cd db
root@2079a8ca822b:/home/workspace/VulnWebApp/Site/db# more __init__.py
import sqlite3
import psycopg2
import random
import string

class Conn_postgres():
    conn = None
    cursor = None

    def __init__(self):
        self.open()

    def open(self):
        self.conn = psycopg2.connect(user = "vulnwebsiteuser",
                                     password = "weakpasswordrule",
                                     host = "localhost",
                                     port = "5432",
                                     database = "vulnwebsite")

        self.cursor = self.conn.cursor()
```

Recommendations:

- ✓ [https://owasp.org/www-community/vulnerabilities/Use of hard-coded password](https://owasp.org/www-community/vulnerabilities/Use%20of%20hard-coded%20password)

Best Practice:

- **All environment should be the same** - By keeping all of your environments the same you will be able to find any issues before they make it out to production.
- **Limit components** - You should push to limit the different components used to only what is really needed, then this will reduce your risk and amount of work to make sure everything is updated and working correctly.
- **Automate Environment creation and validation** - Using automation to create all environments will help reduce user errors and make sure all environments are built exactly the same, next you should automate the validation of the environment to help you understand the health and state of your environments.

VWA Security Report

VWAYMMDD## - A3:2017 - Sensitive Data Exposure - MEDIUM

Vulnerability Exploited: A3-Sensitive Data Exposure

Severity: [Medium]

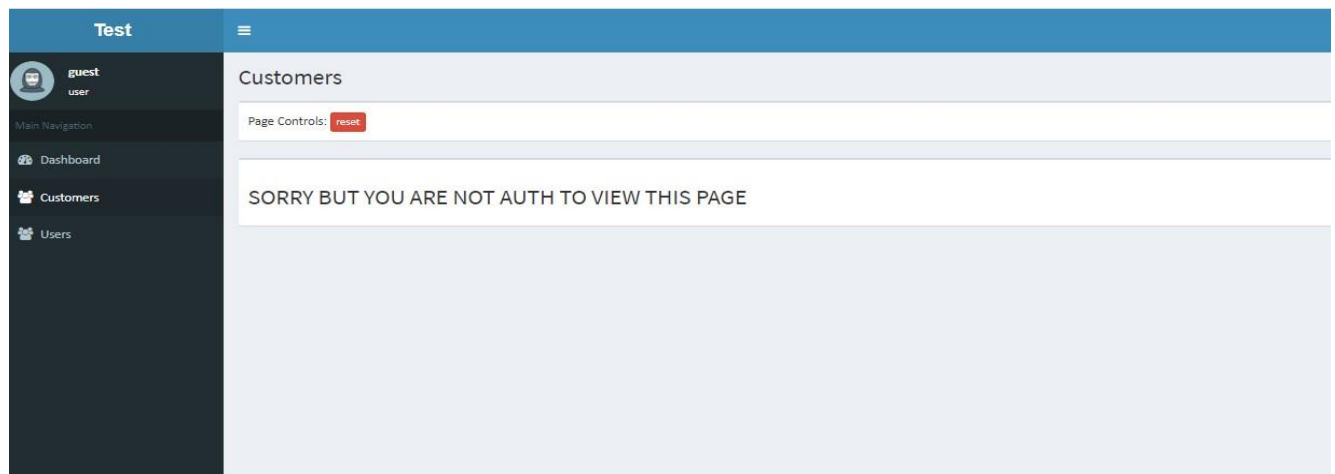
System: VWA Web Application

Vulnerability Explanation:

Sensitive data exposure arises when an application, company, or other entity unintentionally exposes personal data. Many web applications and APIs do not properly protect sensitive data, such as financial, healthcare, and PII. An attacker targets those data to steal or modify to conduct mischief like credit card fraud, identity theft, or other crimes.

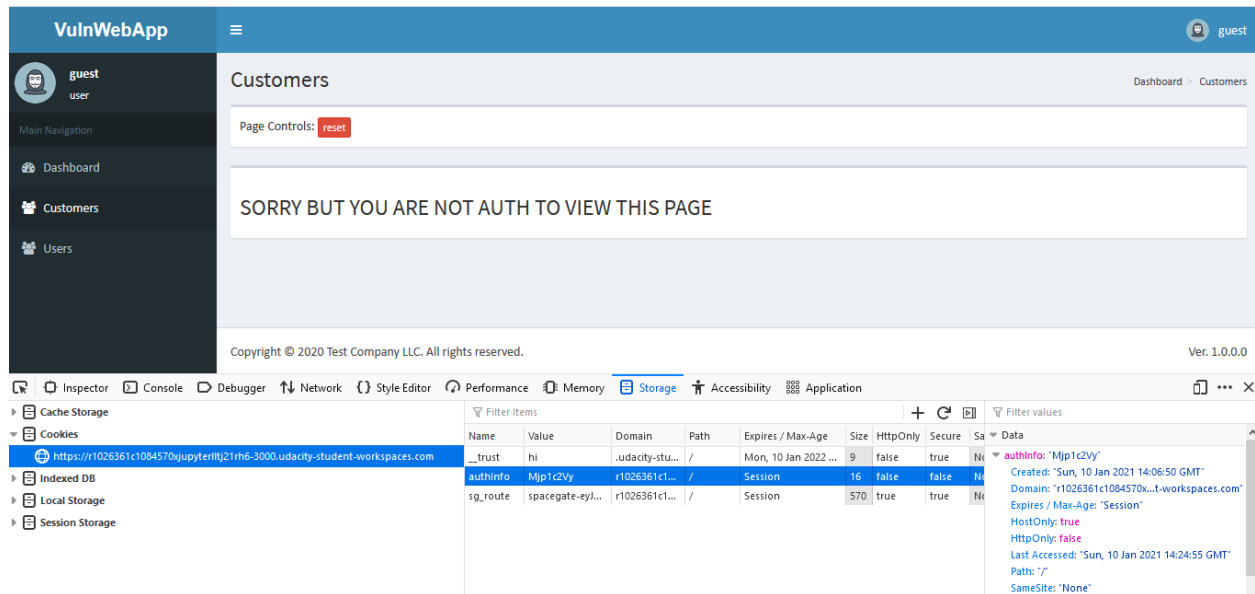
Vulnerability Walk-thru:

1. Go to the customer page



VWA Security Report

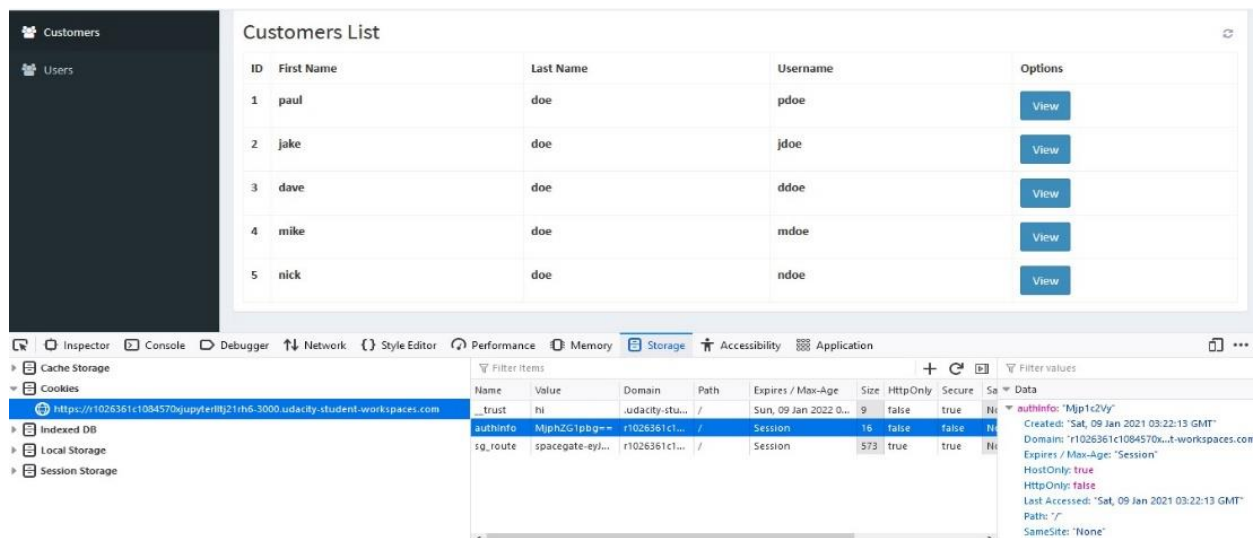
2. Open inspect go to storage then see the authinfo.



The screenshot shows the VulnWebApp interface. The top navigation bar includes a 'guest user' profile and a 'Customers' link. The main content area displays a 'SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE' message. The browser's developer tools are open, showing the 'Storage' tab with a table of cookies. The 'authinfo' cookie is highlighted, showing its details.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	Sa	Data
__trust	hi	.udacity-stu...	/	Mon, 10 Jan 2022 ...	9	false	true	Ni	
authinfo	Mjp1c2Vy	r1026361c1...	/	Session	16	false	false	Ni	Created: "Sun, 10 Jan 2021 14:06:50 GMT" Domain: "r1026361c1084570x...t-workspaces.com" Expires / Max-Age: "Session" HostOnly: true HttpOnly: false Last Accessed: "Sun, 10 Jan 2021 14:24:55 GMT" Path: "/" SameSite: "None"
sg_route	spacegate-eyl...	r1026361c1...	/	Session	570	true	true	Ni	

3. Change the authinfo value user to admin to view this page



The screenshot shows the VulnWebApp interface. The top navigation bar includes a 'Customers' link and a 'Users' link. The main content area displays a 'Customers List' table with columns: ID, First Name, Last Name, Username, and Options. The 'authinfo' cookie is highlighted in the browser's developer tools, showing its details.

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View
4	mike	doe	mdoe	View
5	nick	doe	ndoe	View

VWA Security Report

4. From customer page by viewing the information in developer tools network tab we can see the id=1

Customers List

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

Network Tab: GET /customers/id/?_id=1610292649783 (jQuery.js:9837 (xhr))

Response (JSON):

```
[{"id": 1, "first_name": "paul", "last_name": "doe", "username": "pdoe"}, {"id": 2, "first_name": "jake", "last_name": "doe", "username": "jdoe"}, {"id": 3, "first_name": "dave", "last_name": "doe", "username": "ddoe"}]
```

5. From customer page by viewing the information in developer tools network tab we can see the id=2

Customers List

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

Network Tab: GET /customers/id/?_id=1610292649784 (jQuery.js:9837 (xhr))

Response (JSON):

```
[{"id": 2, "first_name": "jake", "last_name": "doe", "username": "jdoe"}, {"id": 1, "first_name": "paul", "last_name": "doe", "username": "pdoe"}, {"id": 3, "first_name": "dave", "last_name": "doe", "username": "ddoe"}]
```

VWA Security Report

6. From customer page by viewing the information in developer tools network tab we can see the id=3



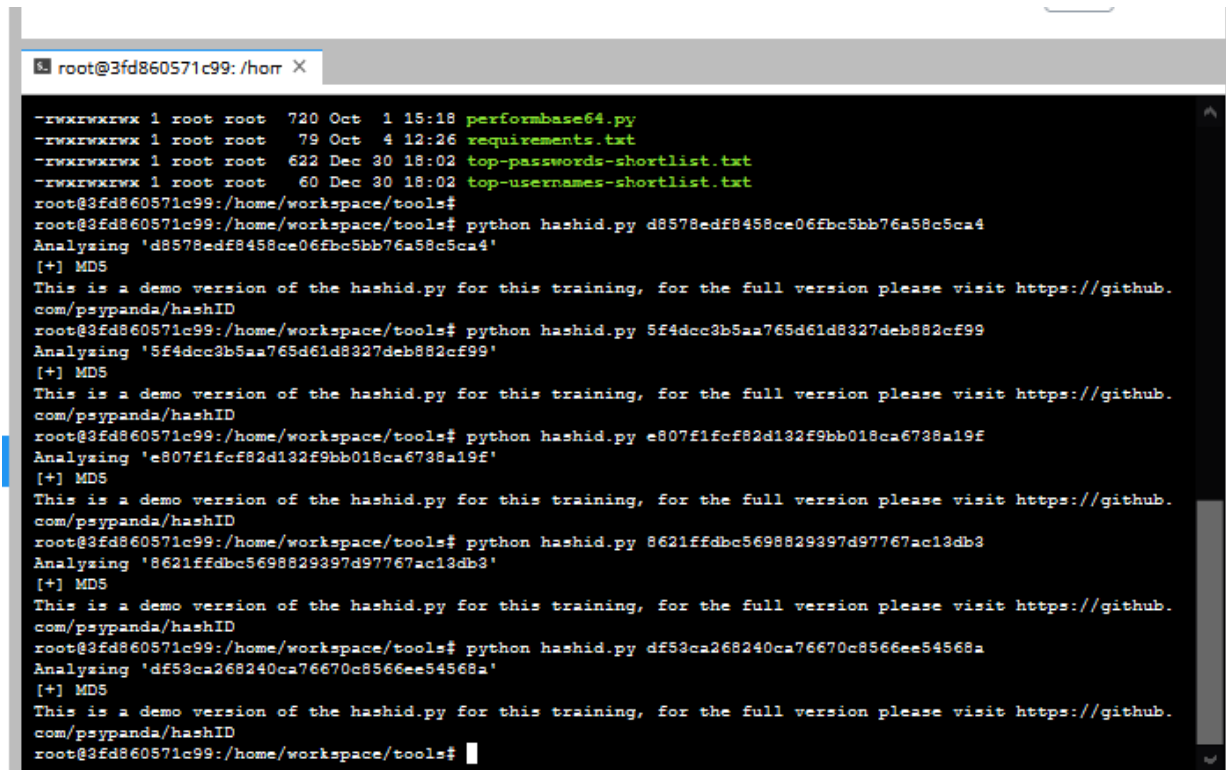
7. From customer page by viewing the information in developer tools network tab we can see the id=4



VWA Security Report

8. Cheking the hash type using hashid.py in workspace

```
python hashid.py d8578edf8458ce06fbc5bb76a58c5ca4
python hashid.py 5f4dcc3b5aa765d61d8327deb882cf99
python hashid.py e807f1fcf82d132f9bb018ca6738a19f
python hashid.py 8621ffdbc5698829397d97767ac13db3
python hashid.py df53ca268240ca76670c8566ee54568a
```



```
root@3fd860571c99: /horr X
-rwxrwxrwx 1 root root 720 Oct 1 15:18 performbase64.py
-rwxrwxrwx 1 root root 79 Oct 4 12:26 requirements.txt
-rwxrwxrwx 1 root root 622 Dec 30 18:02 top-passwords-shortlist.txt
-rwxrwxrwx 1 root root 60 Dec 30 18:02 top-usernames-shortlist.txt
root@3fd860571c99:/home/workspace/tools$
root@3fd860571c99:/home/workspace/tools$ python hashid.py d8578edf8458ce06fbc5bb76a58c5ca4
Analyzing 'd8578edf8458ce06fbc5bb76a58c5ca4'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psyypanda/hashID
root@3fd860571c99:/home/workspace/tools$ python hashid.py 5f4dcc3b5aa765d61d8327deb882cf99
Analyzing '5f4dcc3b5aa765d61d8327deb882cf99'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psyypanda/hashID
root@3fd860571c99:/home/workspace/tools$ python hashid.py e807f1fcf82d132f9bb018ca6738a19f
Analyzing 'e807f1fcf82d132f9bb018ca6738a19f'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psyypanda/hashID
root@3fd860571c99:/home/workspace/tools$ python hashid.py 8621ffdbc5698829397d97767ac13db3
Analyzing '8621ffdbc5698829397d97767ac13db3'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psyypanda/hashID
root@3fd860571c99:/home/workspace/tools$ python hashid.py df53ca268240ca76670c8566ee54568a
Analyzing 'df53ca268240ca76670c8566ee54568a'
[+] MD5
This is a demo version of the hashid.py for this training, for the full version please visit https://github.com/psyypanda/hashID
root@3fd860571c99:/home/workspace/tools$
```

VWA Security Report

9. Using hash cracking from lesson exercise got the passwords.

1. If we give a close look, we can see this is the first 5 letter from of first row. This type of password is easy to guess need to avoid using this.

Hash Cracking

Hash

Type

MD5

▼

Process

Algorithm: MD5
Plaintext: qwerty

This data is provided by hashes.org

2. Commonly used password to attack.

Hash Cracking

Hash

Type

MD5

▼

Process

Algorithm: MD5
Plaintext: password

This data is provided by hashes.org

VWA Security Report

3. This password is first 10 digit from number row. Can be cracked by easy guess or trial and error.

Hash Cracking

Hash

Type

MD5

Process

Algorithm: MD5
Plaintext: 1234567890

This data is provided by hashes.org

4. Brute force may crack this password using common password.

<div><h3>Hash Cracking</h3><div>Hash <input type="text" value="8621ffdbc5698829397d97767ac13db3"/></div><div>Type <div>MD5</div></div><div>Process</div><div>Algorithm: MD5 Plaintext: dragon</div><div>This data is provided by hashes.org</div></div>	<div><h3>Hash Cracking</h3><div>Hash <input type="text" value="df53ca268240ca76670c8566ee54568a"/></div><div>Type <div>MD5</div></div><div>Process</div><div>Algorithm: MD5 Plaintext: computer</div><div>This data is provided by hashes.org</div></div>
---	---

VWA Security Report

Recommendations:

- ✓ <https://cheatsheetseries.owasp.org/cheatsheets/Password Storage Cheat Sheet.html>
- ✓ <https://cheatsheetseries.owasp.org/cheatsheets/User Privacy Protection Cheat Sheet.html>

Best Practice:

- **Classify data** - Classify data processed, stored or transmitted by an application. Identify which data is sensitive according to privacy laws, regulatory requirements, or business needs. Apply controls as per the classification.
- **Encryption:** Make sure to encrypt all sensitive data at rest. Encrypt all data in transit with secure protocols such as TLS with perfect forward secrecy (PFS) ciphers, cipher prioritization by the server, and secure parameters. Enforce encryption using directives like HTTP Strict Transport Security (HSTS)
- **Strengthen Password** - Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt or PBKDF2. Salt and Pepper will increase the strength of your user passwords and make it very difficult to crack.

VWA Security Report

VWAYYMMDD## - A1:2017 - Injection - MEDIUM

Vulnerability Exploited: A1 - Injection
(customer)

Severity: [Medium]

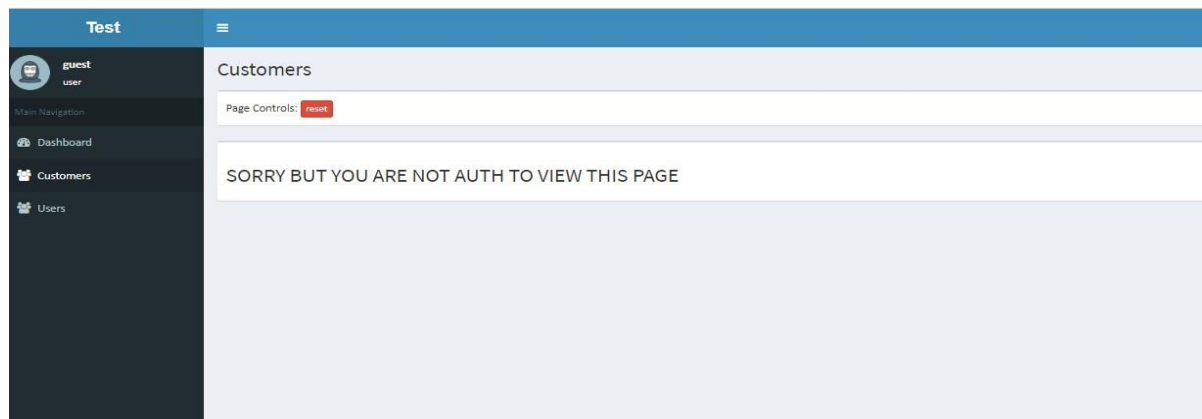
System: VWA Web Application

Vulnerability Explanation:

A1-injection occur when untrusted data is sent to an interpreter as part of a command or query. Injection flaws are SQL, NoSQL, OS, and LDAP. We found out SQL injection in VulWebApp. An attacker can exploit the application by using one of the most common SQL Injection commands. An attacker must first breakout of the current SQL query which will then trick the Database into executing the malicious code.

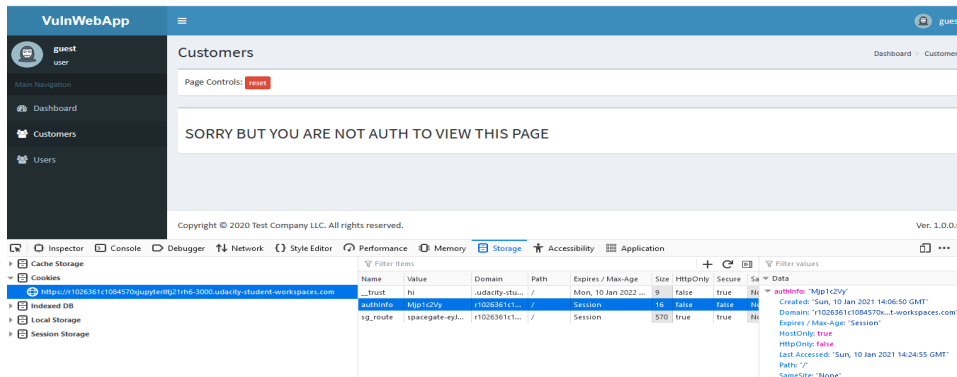
Vulnerability Walk-thru:

1. Go to the customer page.

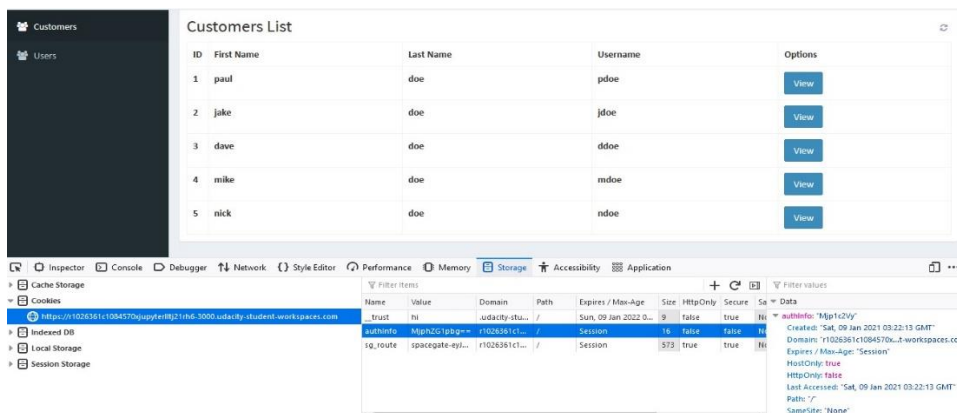


VWA Security Report

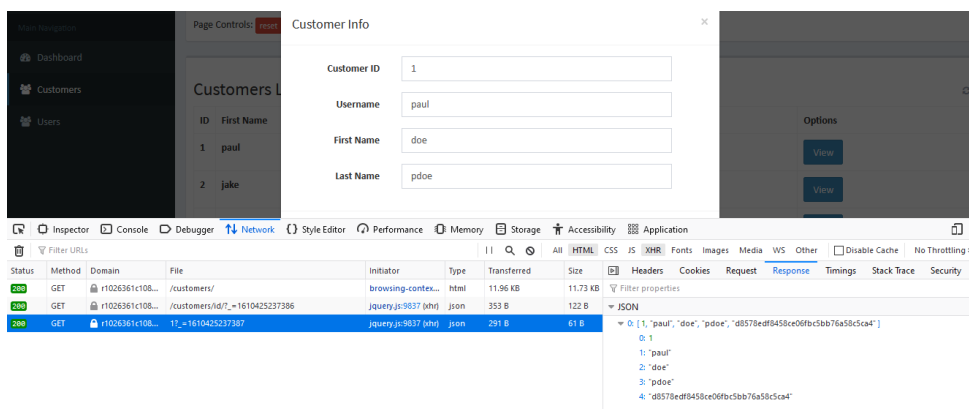
2. Open inspect go to storage then see the authinfo.



3. Change the authinfo value user to admin to view this page



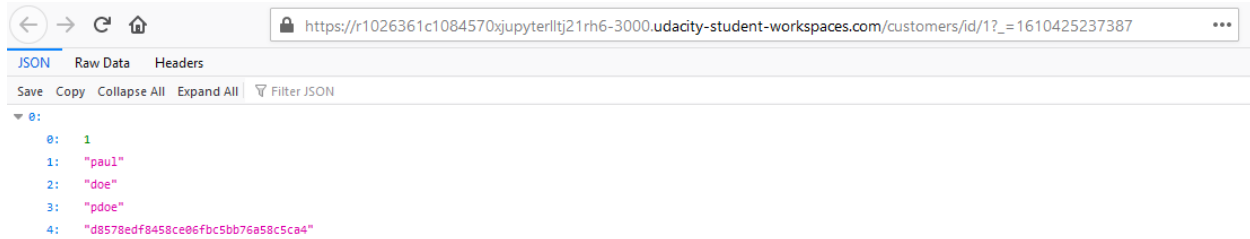
4. Open the inspect and click on network tab the click on view to see customer information.



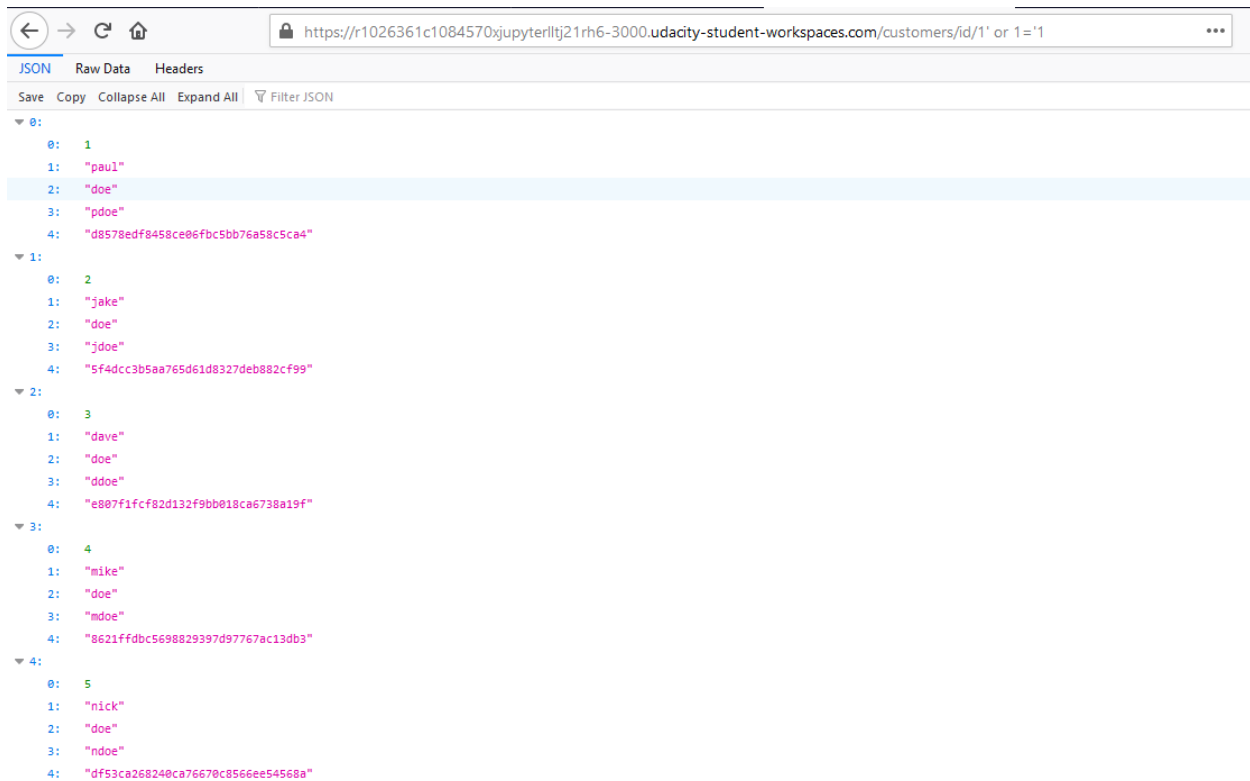
VWAYMMDD - This document is confidential and for internal use only.

VWA Security Report

5. Open the customer id=1 information from network in new tab.



6. Using the most common SQL `` or 1='1' injection in the opened tab of customer url found the information of all customer.



VWA Security Report

Recommendations:

<https://cheatsheetseries.owasp.org/cheatsheets/SQL Injection Prevention Cheat Sheet.html>

Best Practice:

- **Use Parameterized Queries** - This is the best method in preventing SQL Injection, because all variables are limited to the data type which will prevent malicious code from breaking out of the SQL code and preventing the malicious code from running.
- **Sanitize all inputs** - I recommend that you always sanitize all inputs before using them, this will prevent malicious code from running not only in SQL Queries but also in other parts of your code.

VWA Security Report

VWAYMMDD## - A1:2017 - Injection - MEDIUM

Vulnerability Exploited: A1 - Injection (profile)

Severity: [Medium]

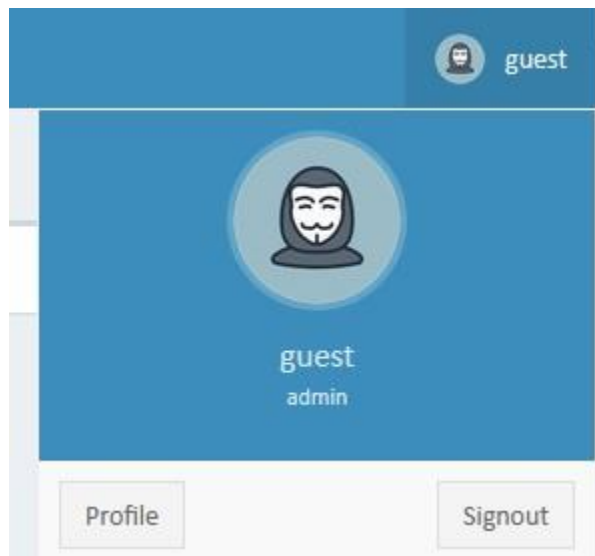
System: VWA Web Application

Vulnerability Explanation:

A1-injection occur when untrusted data is sent to an interpreter as part of a command or query. Injection flaws are SQL, NoSQL, OS, and LDAP. We found out SQL injection in VulWebApp. An attacker can exploit the application by using one of the most common SQL Injection commands. An attacker must first breakout of the current SQL query which will then trick the Database into executing the malicious code.

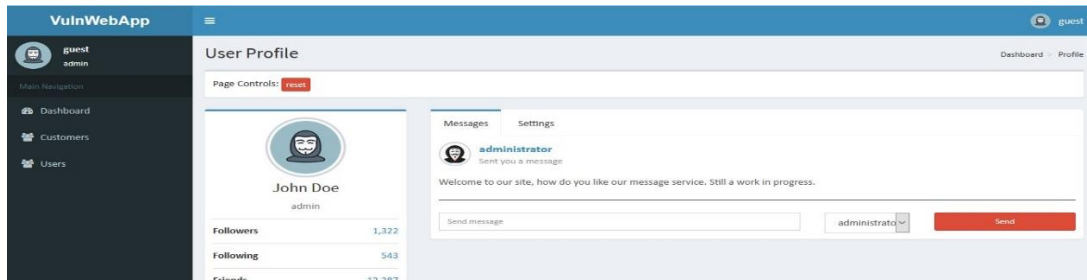
Vulnerability Walk-thru:

1. Go to the profile page as admin.

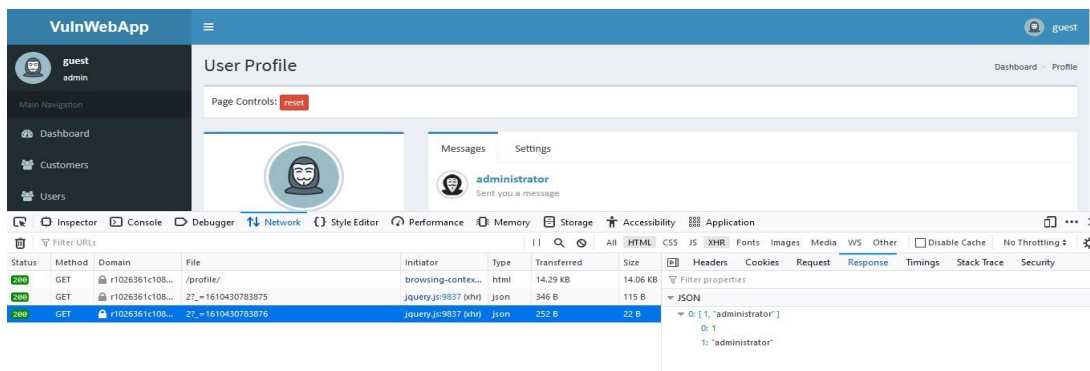


2. Profile page of admin

VWA Security Report



3. Open the inspect and reload the network tab to see userlist.



4. Open then selected inspect in new tab.



5. To see the number of columns inject the SQL injection
"-2` ORDER BY 2--"

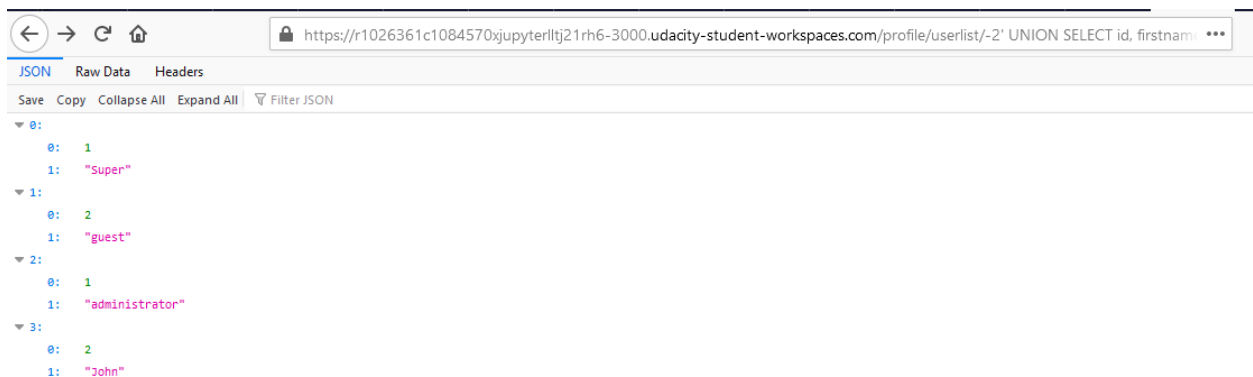


VWA Security Report

6. Again inject the same SQL increasing the value from 2 to 3
"-2' ORDER BY 3--". Here error is showing which means there is only two user value in userlist.

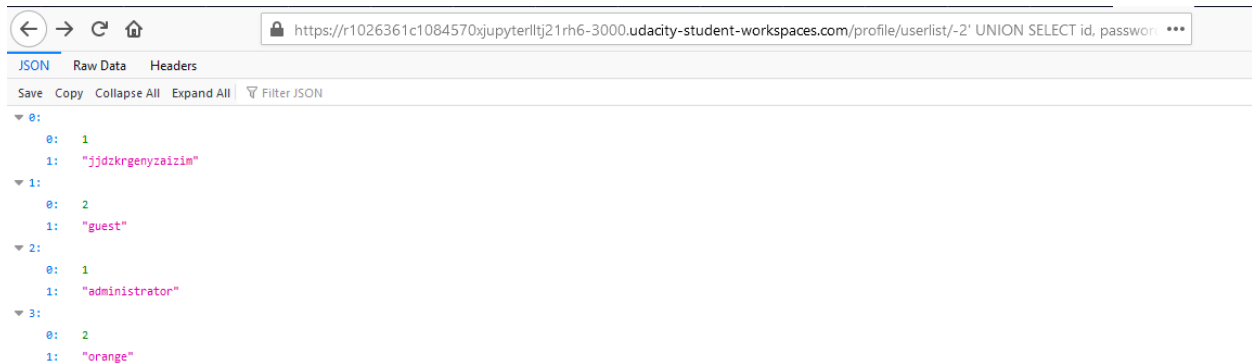


7. To get the details by changing parameters we will inject -2' UNION SELECT id, Firstname FROM users WHERE username!='0. Here username!='0 means need to show both users details . All details of both guest and administrator will be changed from Firstname parameter to Lastname and username parameter.



VWA Security Report

8. To get the password from the details we will put password in first name place. inject -2' UNION SELECT id, password FROM users WHERE username!='0. For both guest and administrator, we got the password parameter.



Recommendations:

[https://cheatsheetseries.owasp.org/cheatsheets/SQL Injection Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html)

Best Practice:

- **Use Parameterized Queries** - This is the best method in preventing SQL Injection, because all variables are limited to the data type which will prevent malicious code from breaking out of the SQL code and preventing the malicious code from running.
- **Sanitize all inputs** - I recommend that you always sanitize all inputs before using them, this will prevent malicious code from running not only in SQL Queries but also in other parts of your code.

VWA Security Report

VWAYYMMDD## - A7:2017 - Cross-Site Scripting (XSS) - MEDIUM

Vulnerability Exploited: A7-Cross-Site Scripting (XSS)

Severity: [Medium]

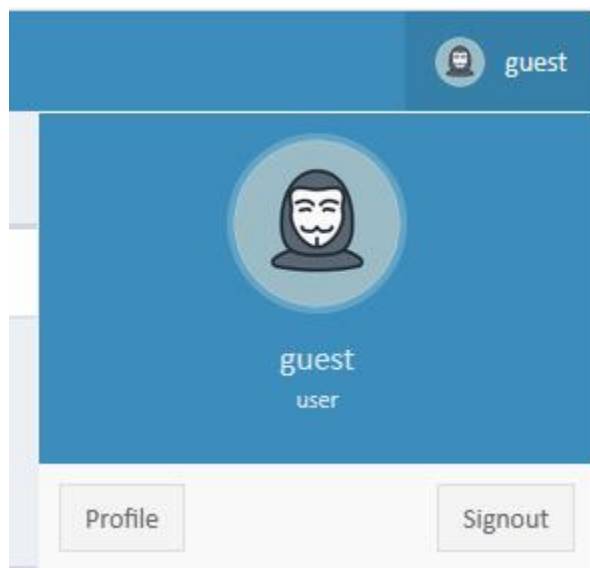
System: VWA Web Application

Vulnerability Explanation:

Cross-Site Scripting (XSS) is used to run unintended code on a victim's local machine it permits attackers to perform scripts in the victim's browser which can hijack user sessions, deface web sites, or redirect the user to malicious sites

Vulnerability Walk-thru:

1. Go to the profile page

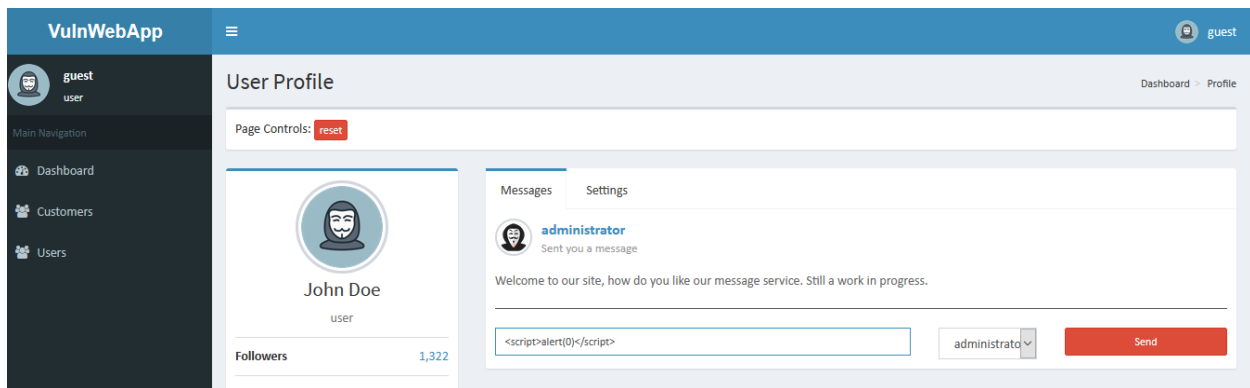


VWA Security Report

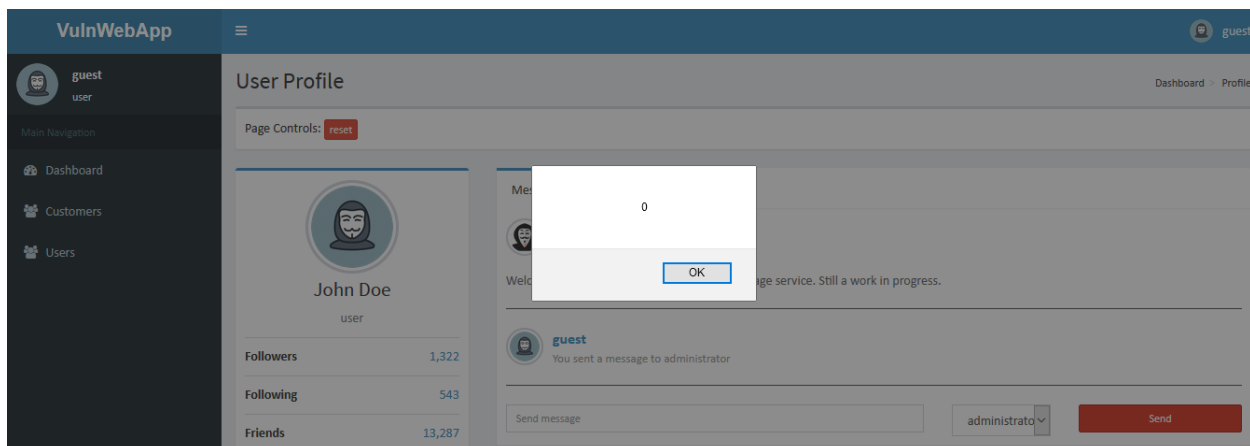
Stored XSS

In this type of attack, a payload is sent to the web application where it is saved in the DB and then when called from the database it is executed on the target local machine

2. Injecting `<script>alert(0)</script>` in send message option



3. Showing the executed alert in profile page

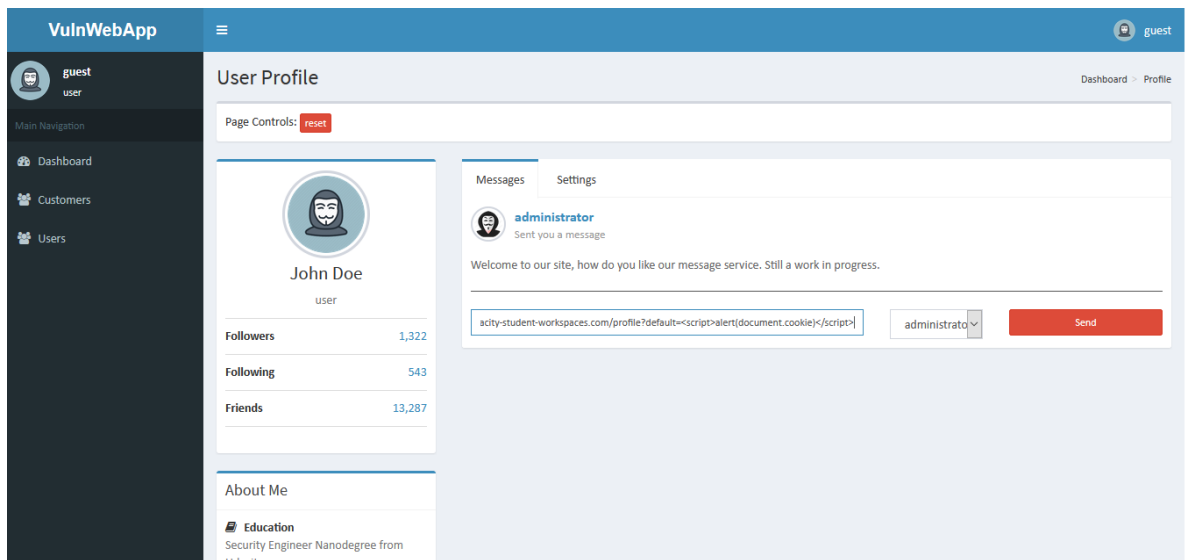


VWA Security Report

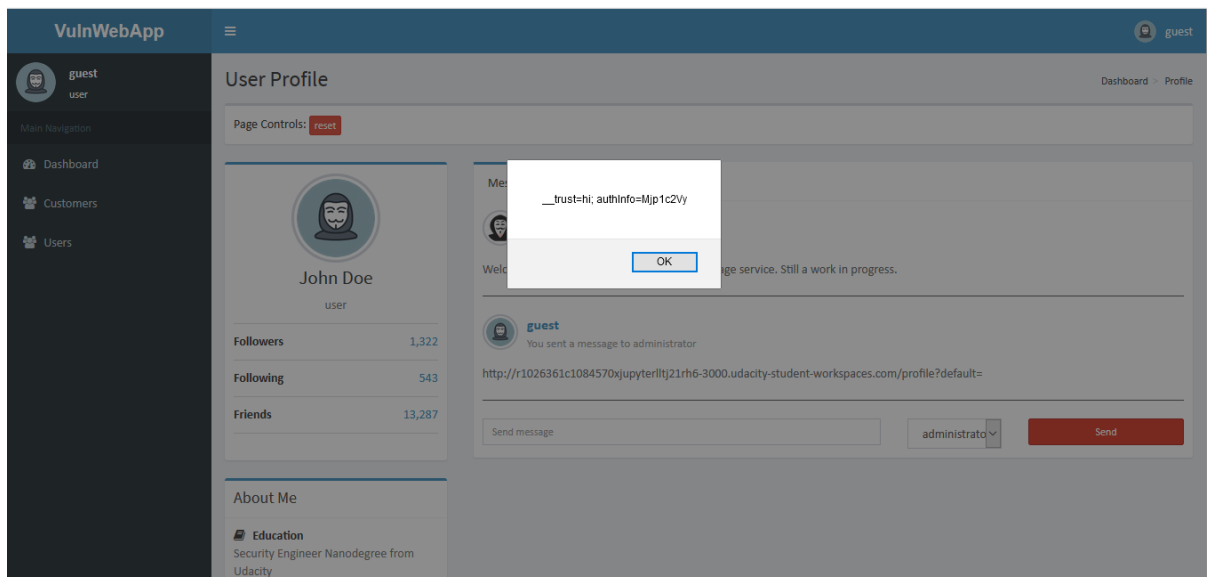
DOM XSS:

This type of attack is considered a more advanced type of XSS and is when the vulnerability appears up in the Document Object Model (DOM) rather than in the html page.

4. Injecting [http://r1026361c1084570xjupyterl1tj21rh6-3000.udacity-student-workspaces.com/profile?default=<script>alert\(document.cookie\)</script>](http://r1026361c1084570xjupyterl1tj21rh6-3000.udacity-student-workspaces.com/profile?default=<script>alert(document.cookie)</script>)



5. Vulnerability appearing up in document object model.



VWA Security Report

Recommendations:

- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/Cross Site Scripting Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross%20Site%20Scripting%20Prevention%20Cheat%20Sheet.html)
- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/DOM based XSS Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/DOM%20based%20XSS%20Prevention%20Cheat%20Sheet.html)

Best Practice:

- **Using Frameworks** - It automatically escape XSS by design, such as the latest Ruby on Rails, React JS. Learn the limitations of each framework's XSS protection and appropriately handle the use cases which are not covered
- **Sanitizing all inputs** - All input that is entered by the user should be precisely validated, because the user's input may find its way to the output.
- **Filtering inputs** - The idea of the filtering is to search for risky keywords in the user's input and remove them or replace them by empty strings.
- **Characters escaping** - You can use appropriate characters to change them by special codes. There exist appropriate libraries to escape the characters.

VWA Security Report

VWAYMMDD## - A2:2017 - Broken

Authentication - MEDIUM

Vulnerability Exploited: A2-Broken Authentication

Severity: [Medium]

System: VWA Web Application

Vulnerability Explanation:

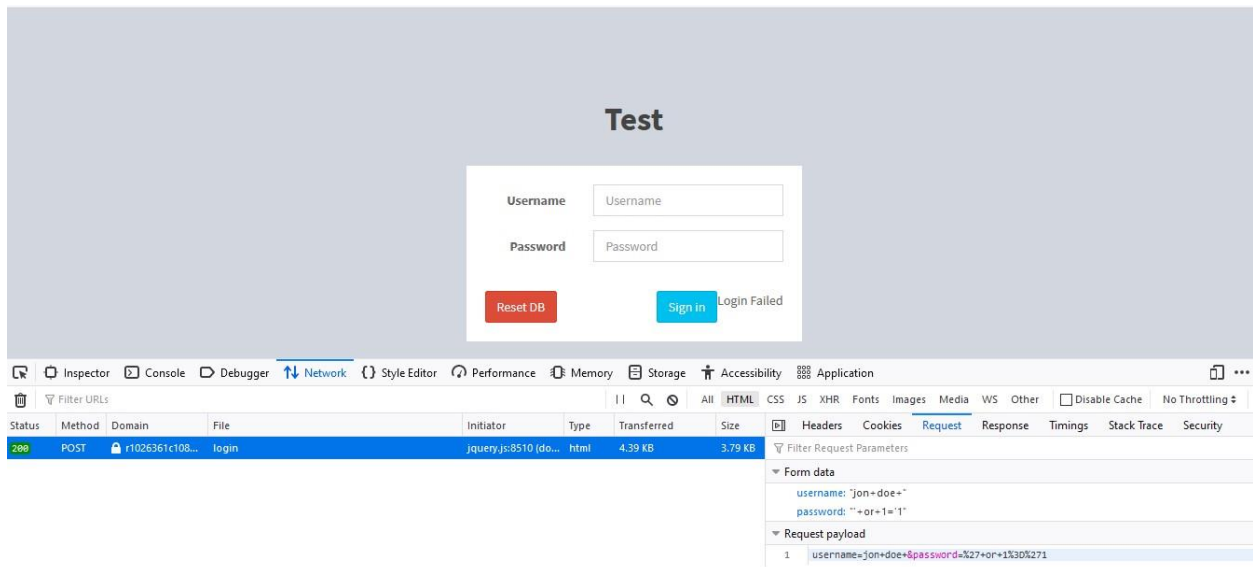
Here brute force method is used in broken authentication. By using large number of common username and password through brute-force login tools against a web application an attacker tries to find the combination from the list.

Vulnerability Walk-thru:

1. For login go to the url:

<https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-student-workspaces.com/login>

2. Credentials for login is unknown and SQL injection is failed

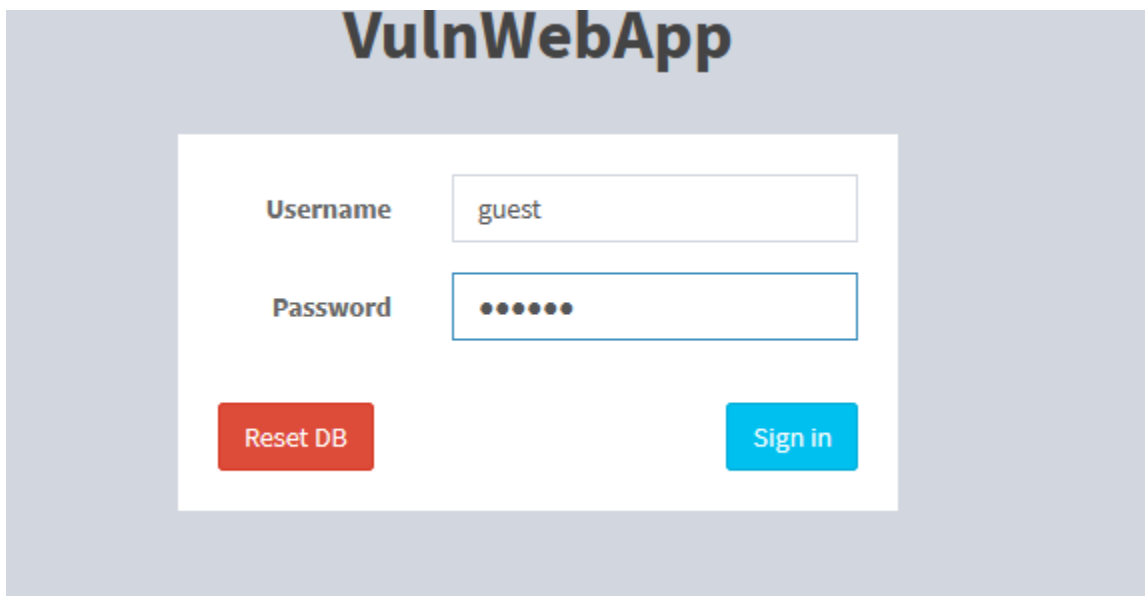


VWA Security Report

3. From tools option in workspace using bruteforce.py to get match credentials for login.
4. `python bruteforce.py -U top-usernames-shortlist.txt -P top-passwords-shortlist.txt -d username=^USR^:password=^PWD^ -m 'post' -f 'Login Failed'`
<https://r1026361c1084570xjupyterlltj21rh6-3000.udacity-student-workspaces.com/login>

```
root@591b91c509c2:/hor x
[-] Login Failed! ('username': 'guest', 'password': 'pupper')
[-] Login Failed! ('username': 'guest', 'password': 'lill')
[-] Login Failed! ('username': 'guest', 'password': 'sacred')
[-] Login Failed! ('username': 'guest', 'password': 'william')
[-] Login Failed! ('username': 'guest', 'password': 'daniel')
[-] Login Failed! ('username': 'guest', 'password': 'galder')
[-] Login Failed! ('username': 'guest', 'password': 'summer')
[-] Login Failed! ('username': 'guest', 'password': 'heathcliff')
[-] Login Failed! ('username': 'guest', 'password': 'summer')
[-] Login Failed! ('username': 'guest', 'password': 'johanna')
[-] Login Failed! ('username': 'guest', 'password': 'heggie')
[-] Login Failed! ('username': 'guest', 'password': 'mattis')
[-] Login Failed! ('username': 'guest', 'password': 'ashley')
[-] Login Failed! ('username': 'guest', 'password': 'thunder')
[-] Login Failed! ('username': 'guest', 'password': 'keweenaw')
[-] Login Failed! ('username': 'guest', 'password': 'silvers')
[-] Login Failed! ('username': 'guest', 'password': 'richard')
[+] Login Found! ('username': 'guest', 'password': 'orange')
This is a demo code used for this training.
root@591b91c509c2:/home/workspace/tools#
```

5. Using bruteforce.py tool got the matched credentials for login where user: guest password: orange



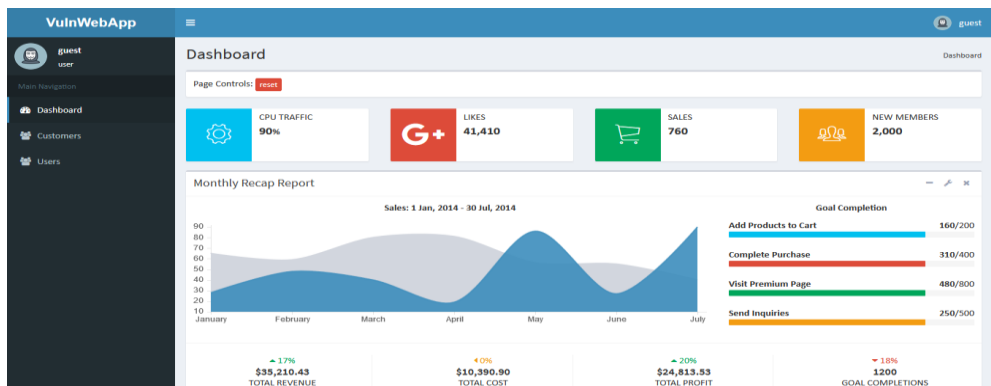
VulnWebApp

Username

Password

VWA Security Report

6. Got the access of test page.



Recommendations:

- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/Credential Stuffing Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Credential_Stuffing_Prevention_Cheat_Sheet.html)
- ✓ <https://github.com/danielmiessler/SecLists/tree/master/Passwords>

Best Practice:

- **Multi-Factor Authentication (MFA)** – MFA prevents automated attacks like credential stuffing, brute force, and stolen credential re-use attacks. So, using MFA would be the best methods for protecting accounts, even if a user credentials are exposed to the internet either by data breaches or stolen using some type of exploit the user account is still safe because the attacker will not be able to login without the MFA.
- **Captcha** – Using a captcha is a good way to slow down or even stop attackers from using some sort of automation on your web application.
- **Weak-password check** – Attempting weak-password check by using top 1000 worst passwords against new or changed password.
- **Failed Login Attempts** – By using failed login attempts you can auto lock accounts or even block IP address that appears to be using some sort of brute forcing against a user account.

VWA Security Report

VWAYMMDD## - A5:2017 - Broken Access Control - MEDIUM

Vulnerability Exploited: A5 - Broken Access Control (customer)

Severity: [Medium]

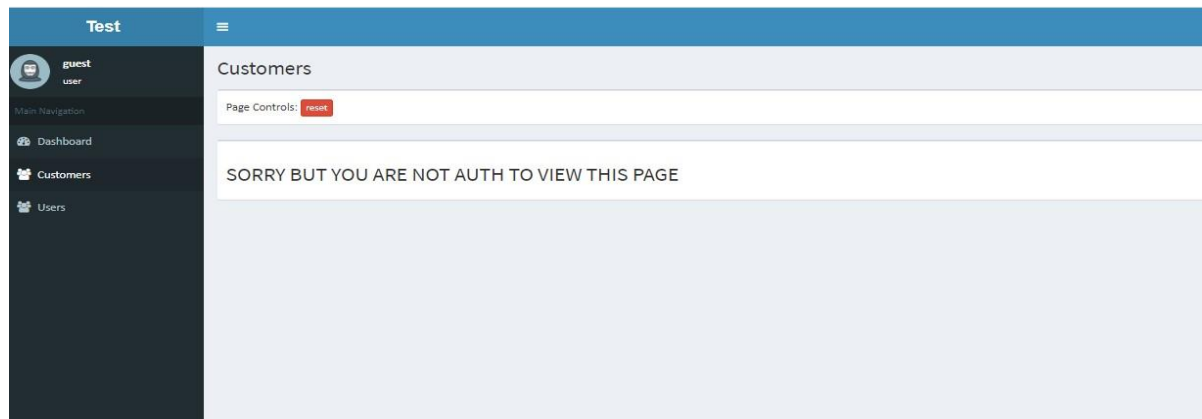
System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

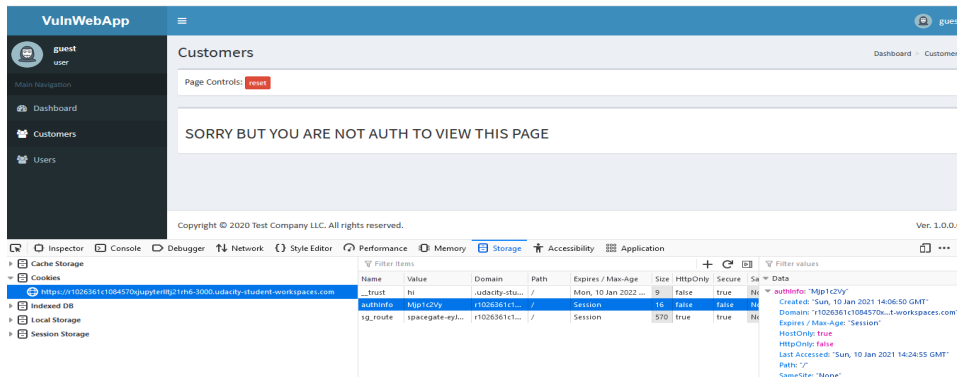
Vulnerability Walk-thru:

1. Go to the customer page.



VWA Security Report

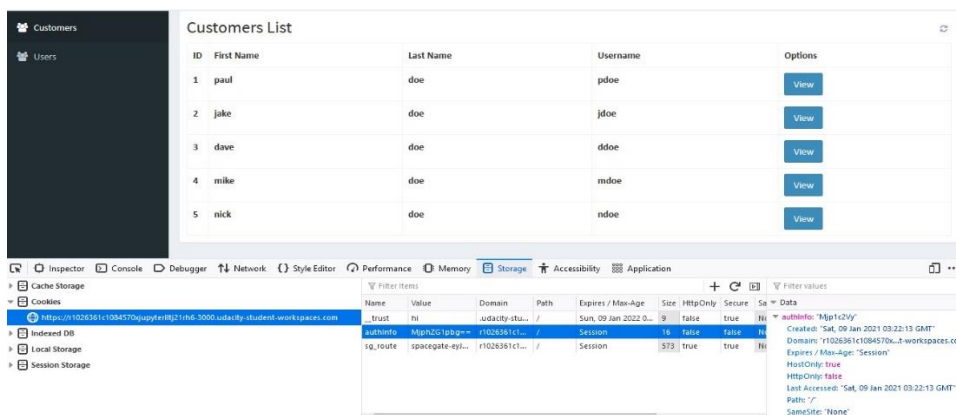
2. Open inspect go to storage then see the authinfo.



3. Decode the given authinfo value then encode the 2:admin value in terminal

```
root@2079a8ca822b: /home/
.1.2 idna-2.10 itsdangerous-1.1.0 pycop2-binary-2.8.6 python-dotenv-0.14.0 requests-2.24.0 six-1.15.0 urllib3-1.25.10
root@2079a8ca822b:/home/workspace# cd /home/workspace/tools && ls -la
total 44
drwxrwxrwx 3 root root 4096 Dec 30 18:14 .
drwxr-xr-x 5 root root 4096 Jan 12 02:45 ..
-rwxrwxrwx 1 root root 3625 Dec 30 18:14 bruteforce.py
-rwxrwxrwx 1 root root 2726 Oct 18 16:03 checkhash.py
-rwxrwxrwx 1 root root 6766 Oct 1 14:50 hashid.py
drwxr-xr-x 2 root root 4096 Dec 28 18:24 .ipynb checkpoints
-rwxrwxrwx 1 root root 720 Oct 1 15:18 performbase64.py
-rwxrwxrwx 1 root root 79 Oct 4 12:26 requirements.txt
-rwxrwxrwx 1 root root 622 Dec 30 18:02 top-passwords-shortlist.txt
-rwxrwxrwx 1 root root 60 Dec 30 18:02 top-usernames-shortlist.txt
root@2079a8ca822b:/home/workspace/tools# python performbase64.py -d Mjplc2Vy
2:user
root@2079a8ca822b:/home/workspace/tools# python performbase64.py 2:admin
MjphZG1pbG==
root@2079a8ca822b:/home/workspace/tools#
```

4. Change the authinfo value 2:user to 2:admin to view this page



VWA Security Report

5. There are 5 customers in customers list with respected id and value

The screenshot shows a web application interface with a sidebar containing 'Customers' and 'Users' links. The main area displays a 'Customers List' table with 5 rows. Each row has a 'View' button. Below the table, a network inspector shows a list of requests. The selected request is a GET request to the URL `/customers/id/?_id=1610341421508` with a response of type 'JSON'.

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	<button>View</button>
2	jake	doe	jdoe	<button>View</button>
3	dave	doe	ddoe	<button>View</button>
4	mike	doe	mdoe	<button>View</button>
5	nick	doe	ndoe	<button>View</button>

Network Inspector Details:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	Headers	Cookies	Request	Response	Timings	Stack Trace	Security
200	GET	r1026361c108...	/customers/	document	html	11.96 KB	11.73 KB							
200	GET	r1026361c108...	/customers/id/?_id=1610341421508	jquery.js:9837 (xhr)	json	353 B	122 B				[{"id": 1, "first_name": "paul", "last_name": "doe", "username": "pdoe", "password": "d8578edf8458ce06fbc5bb76a58c5ca4"}]			
200	GET	r1026361c108...	2?_id=1610341421509	jquery.js:9837 (xhr)	json	291 B	61 B				[{"id": 2, "first_name": "jake", "last_name": "doe", "username": "jdoe", "password": "5f4dccc3b5a8765d61d8327deb882cf99"}]			
200	GET	r1026361c108...	3?_id=1610341421510	jquery.js:9837 (xhr)	json	291 B	61 B				[{"id": 3, "first_name": "dave", "last_name": "doe", "username": "ddoe", "password": "d8578edf8458ce06fbc5bb76a58c5ca4"}]			
200	GET	r1026361c108...	4?_id=1610341421511	jquery.js:9837 (xhr)	json	291 B	61 B				[{"id": 4, "first_name": "mike", "last_name": "doe", "username": "mdoe", "password": "d8578edf8458ce06fbc5bb76a58c5ca4"}]			
200	GET	r1026361c108...	5?_id=1610341421512	jquery.js:9837 (xhr)	json	291 B	61 B				[{"id": 5, "first_name": "nick", "last_name": "doe", "username": "ndoe", "password": "d8578edf8458ce06fbc5bb76a58c5ca4"}]			

6. Information of customer where id=1

The screenshot shows a web browser address bar with the URL `https://r1026361c1084570xjupyterl1tj21rh6-3000.udacity-student-workspaces.com/customers/id/1?_id=1610341421508`. The JSON response is displayed in the console:

```
{
  "id": 1,
  "first_name": "paul",
  "last_name": "doe",
  "username": "pdoe",
  "password": "d8578edf8458ce06fbc5bb76a58c5ca4"
}
```

7. Information of customer where id=2 is showing by changing the id number in url instead of viewing it from customer page.

The screenshot shows a web browser address bar with the URL `https://r1026361c1084570xjupyterl1tj21rh6-3000.udacity-student-workspaces.com/customers/id/2?_id=1610341421508`. The JSON response is displayed in the console:

```
{
  "id": 2,
  "first_name": "jake",
  "last_name": "doe",
  "username": "jdoe",
  "password": "5f4dccc3b5a8765d61d8327deb882cf99"
}
```

VWA Security Report

8. Information of customer where id=3 is showing by changing the id number in url instead of viewing it from customer page.



9. Information of customer where id=4 is showing by changing the id number in url instead of viewing it from customer page.



10. Information of customer where id=5 is showing by changing the id number in url instead of viewing it from customer page.



11. As there are 5 customers so there exist no information for id=6



VWA Security Report

Recommendations:

- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/Access Control Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html)
- ✓ <https://owasp.org/www-project-application-security-verification-standard/>

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages.
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

VWAYYMMDD## - A5:2017 - Broken Access

Control - MEDIUM

Vulnerability Exploited: A5 - Broken Access Control
(profile)

Severity: [Medium]

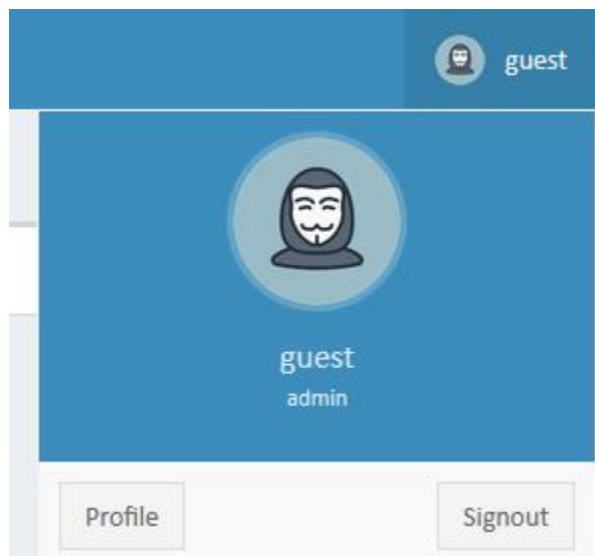
System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

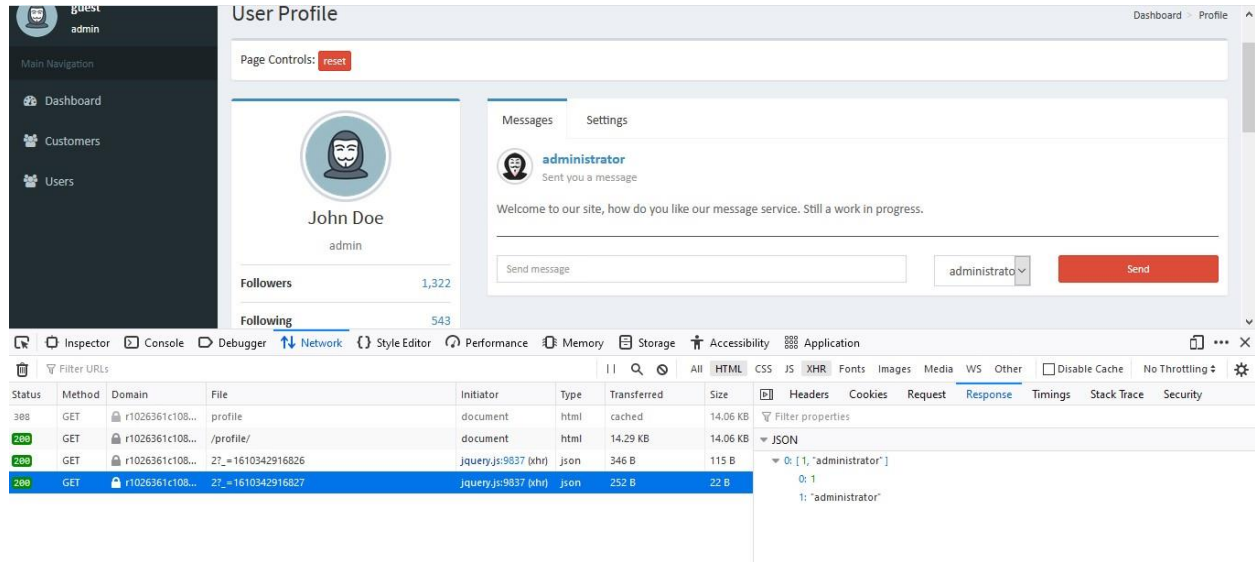
Vulnerability Walk-thru:

1. Go to profile as admin



VWA Security Report

2. Open the inspect. There are 2 userlists in profile one is administration another is guest.



3. Open the userlist in new tab where id =2 is showing the information of administration

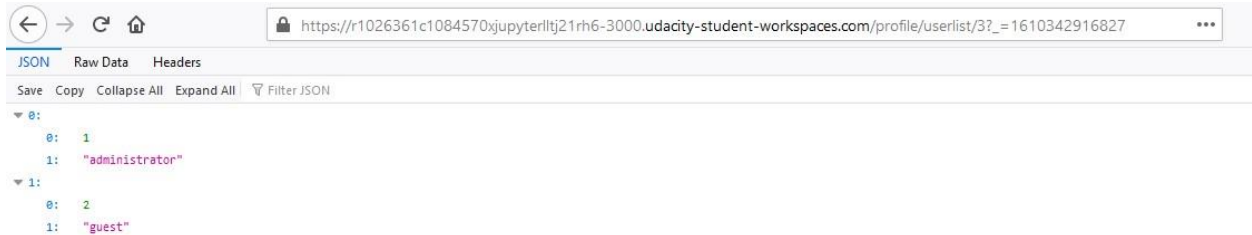


4. Information of userlist where id=1 is showing the information by changing the id number in url instead of viewing it from profile page.



VWA Security Report

5. As there are only two userlist where id=3 is showing both guest and administration. Whatever we use except id=1 and id=2 it shows both guest and administration information.



Recommendations:

- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/Access Contr ol Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Access_Contr ol Cheat Sheet.html)
- ✓ <https://owasp.org/www-project-application-security-verification-standard/>

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected.
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages.
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

VWAYMMDD## - A5:2017 - Broken Access Control - LOW

Vulnerability Exploited: A5 - Broken Access Control (user)

Severity: [Low]

System: VWA Web Application

Vulnerability Explanation:

Broken access controls often fail to enforce restrictions on authenticated users. An attacker takes this advantage to get unauthorized access to restricted areas of the application such as accessing other users' accounts, viewing sensitive files, modifying other users' data, changing access rights, etc.

Vulnerability Walk-thru:

As we changed the authinfo value to see customer page and got access as admin we also can view both customer and user page.

1. There are 2 users in the users list with respected id and value.

The screenshot displays the VWA application interface. On the left is a sidebar with a user profile (guest/admin) and navigation links for Dashboard, Customers, and Users. The main content area is titled 'Users' and contains a 'Page Controls' section with a 'reset' button. Below this is a 'Users List' table with the following data:

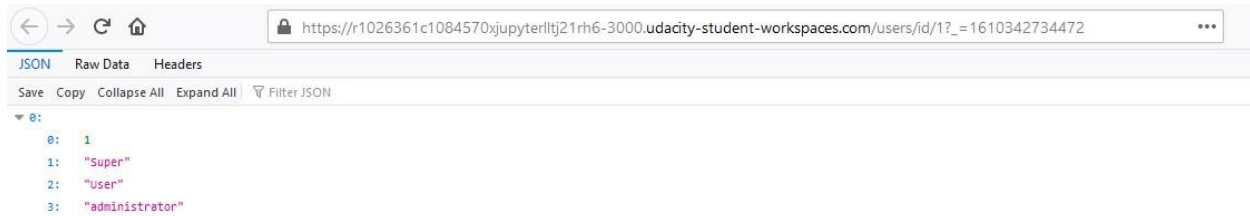
ID	First Name	Last Name	Username	Options
1	Super	User	administrator	View
2	John	Doe	guest	View

At the bottom of the screenshot, the browser's developer tools are open, showing the 'Network' tab. A list of network requests is visible, with the third request (a GET request to a JSON endpoint) selected. The 'Response' pane on the right shows the JSON data returned by the request:

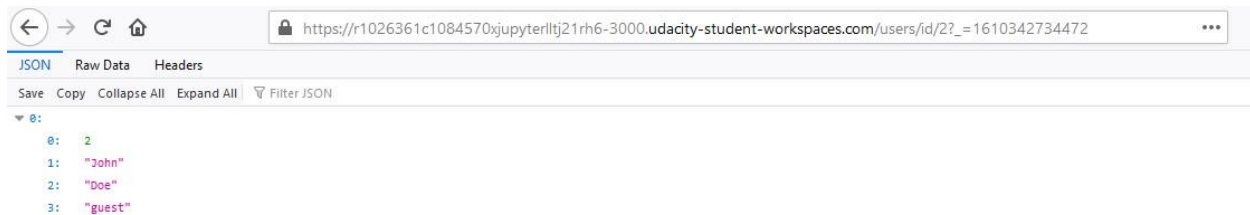
```
[{"id": 1, "first_name": "Super", "last_name": "User", "username": "administrator"}, {"id": 2, "first_name": "John", "last_name": "Doe", "username": "guest"}]
```

VWA Security Report

2.Information of user where id=1



3. Information of user where id=2 is showing by changing the id in url instead of viewing it from user page.



4. As there are 2 users so information is available for id=3.



VWA Security Report

Recommendations:

- ✓ [https://cheatsheetseries.owasp.org/cheatsheets/Access Contr ol Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Access_Contr ol Cheat Sheet.html)
- ✓ <https://owasp.org/www-project-application-security-verification-standard/>

Best Practice:

- **Rate Limit Data** - by rate limiting the user access to data on the site, you can slow down their ability to scrape all data from your web application.
- **Re-validate on all secure pages** - make sure you have proper testing around all secure pages and endpoints to validate access control are working as expected.
- **Deny Access for non-public pages by default** - you should have general rules that auto deny access to non-public pages and require validation to access these pages
- **Log Access failures** - you should not only log all access failures, but also create automation process that alert you of IP/Users that have high level of failures.

VWA Security Report

**VWAYMMDD## - A8:2017 - Insecure
Deserialization - LOW**

Vulnerability Exploited: A8 - Insecure Deserialization
Exercise Solution

Severity: [Low]

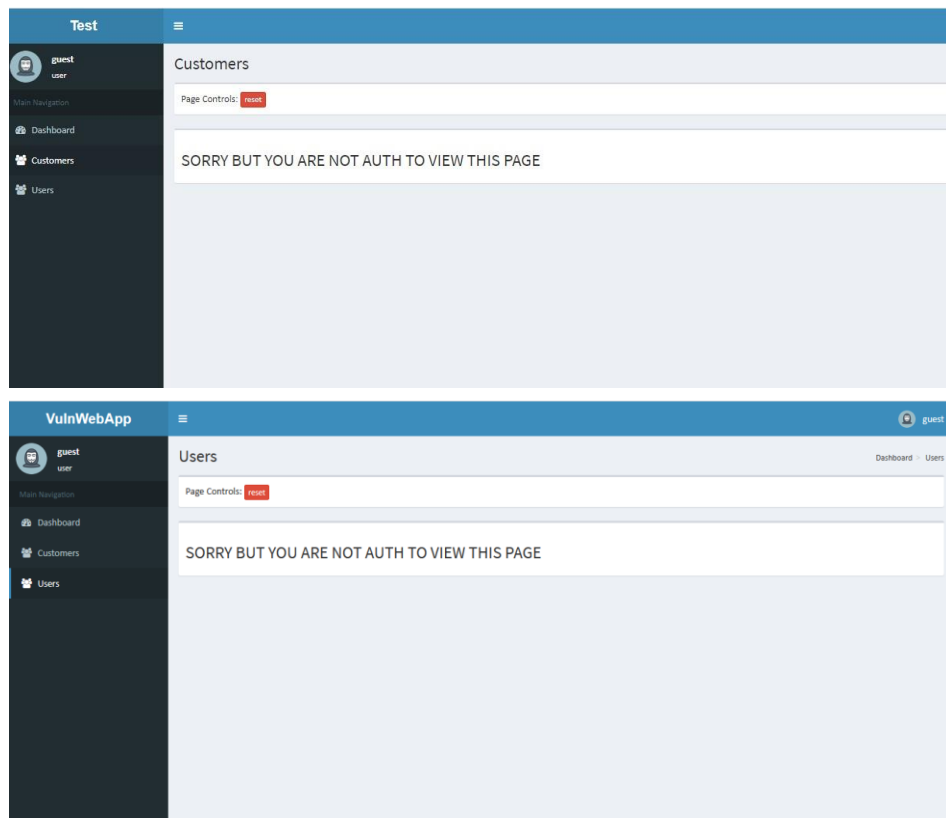
System: VWA Web Application

Vulnerability Explanation:

Insecure deserialization often leads to remote code execution by trusting a data source without validating. An attacker can modify their access level and view area of the site they were not intended to see.

Vulnerability Walk-thru:

1. Access denied for user to see customers and users list.



VWA Security Report

2.To know about the access level open the developer tool go to the storage click on the cookies and see the value of authinfo.

The screenshot shows the VulnWebApp interface. The top navigation bar includes 'VulnWebApp' and a user profile 'guest user'. The main content area is titled 'Customers' and displays a message: 'SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE'. Below this, a copyright notice 'Copyright © 2020 Test Company LLC. All rights reserved.' and version 'Ver. 1.0.0.0' are visible. The bottom section shows the browser's developer tools, specifically the 'Storage' tab. A table lists cookies, with the 'authinfo' cookie highlighted. The 'authinfo' cookie has a value of 'Mjplc2Vy' and is associated with the domain 'r1026361c1084570x...t-workspaces.com'. The 'Data' column for this cookie shows its details: Created: 'Sun, 10 Jan 2021 14:06:50 GMT', Domain: 'r1026361c1084570x...t-workspaces.com', Expires / Max-Age: 'Session', HostOnly: true, HttpOnly: false, Last Accessed: 'Sun, 10 Jan 2021 14:24:55 GMT', Path: '/', and SameSite: 'None'.

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	Sa	Data
__trust	hi	.udacity-stu...	/	Mon, 10 Jan 2022 ...	9	false	true	Nc	
authinfo	Mjplc2Vy	r1026361c1...	/	Session	16	false	false	Nc	Created: "Sun, 10 Jan 2021 14:06:50 GMT" Domain: "r1026361c1084570x...t-workspaces.com" Expires / Max-Age: "Session" HostOnly: true HttpOnly: false Last Accessed: "Sun, 10 Jan 2021 14:24:55 GMT" Path: "/" SameSite: "None"
sg_route	spacegate-eyl...	r1026361c1...	/	Session	570	true	true	Nc	

3. Authinfo value is in string so need to decode the value. using performbase64.py file in the terminal to decode value = Mjplc2Vy. As only admin has the permission to get access encoding 2:admin in string to use it in authinfo value.

```
root@2079a8ca822b: /home/

.1.2 idna-2.10 itdangerous-1.1.0 psycpg2-binary-2.8.6 python-dotenv-0.14.0 requests-2.24.0 six-1.15.0 urllib3-1.25.10
root@2079a8ca822b:/home/workspace# cd /home/workspace/tools && ls -la
total 44
drwxrwxrwx 3 root root 4096 Dec 30 18:14 .
drwxr-xr-x 5 root root 4096 Jan 12 02:45 ..
-rwxrwxrwx 1 root root 3625 Dec 30 18:14 bruteforce.py
-rwxrwxrwx 1 root root 2726 Oct 18 16:03 checkhash.py
-rwxrwxrwx 1 root root 6766 Oct 1 14:50 hashid.py
drwxr-xr-x 2 root root 4096 Dec 28 18:24 .ipynb_checkpoints
-rwxrwxrwx 1 root root 720 Oct 1 15:18 performbase64.py
-rwxrwxrwx 1 root root 79 Oct 4 12:26 requirements.txt
-rwxrwxrwx 1 root root 622 Dec 30 18:02 top-passwords-shortlist.txt
-rwxrwxrwx 1 root root 60 Dec 30 18:02 top-usernames-shortlist.txt
root@2079a8ca822b:/home/workspace/tools# python performbase64.py -d Mjplc2Vy
2:user
root@2079a8ca822b:/home/workspace/tools# python performbase64.py 2:admin
MjphZGIpbG==
root@2079a8ca822b:/home/workspace/tools#
```


VWA Security Report

4. Placed the encoded 2:admin value in string MjphZG1pbG== in cookies. Got the access as admin to see customers and users list.

Customers List

ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View
4	mike	doe	mdoe	View
5	nick	doe	ndoe	View

Storage

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Data
__trust	hi	.udacity-stu...	/	Sun, 09 Jan 2022 0...	9	false	true	None	
authinfo	MjphZG1pbG==	r1026361c1...	/	Session	16	false	false	None	Created: 'Sat, 09 Jan 2021 03:22:13 GMT' Domain: 'r1026361c1084570x...t-workspaces.con' Expires / Max-Age: 'Session' HostOnly: true HttpOnly: false Last Accessed: 'Sat, 09 Jan 2021 03:22:13 GMT' Path: '/' SameSite: 'None'
sg_route	spacegate-eyj...	r1026361c1...	/	Session	573	true	true	None	

Users

Page Controls: [reset](#)

Users List

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

Storage

Name	Value	Domain	Path	Expires / Max-Age	Size	HttpOnly	Secure	SameSite	Data
__trust	hi	.udacity-stu...	/	Mon, 10 Jan 2022 ...	9	false	true	None	
authinfo	MjphZG1pbG==	r1026361c1...	/	Session	20	false	false	None	Created: 'Sun, 10 Jan 2021 03:58:33 GMT' Domain: 'r1026361c1084570x...t-workspaces.con' Expires / Max-Age: 'Session' HostOnly: true HttpOnly: false Last Accessed: 'Sun, 10 Jan 2021 03:59:07 GMT' Path: '/' SameSite: 'None'
sg_route	spacegate-eyj...	r1026361c1...	/	Session	568	true	true	None	

VWA Security Report

Recommendations:

- ✓ <https://cheatsheetseries.owasp.org/cheatsheets/Deserialization Cheat Sheet.html>
- ✓ <https://owasp.org/www-project-proactive-controls/v3/en/c5-validate-inputs>

Best Practice

- **Use Serialization Mediums** - The only safe architectural pattern is not to accept serialized objects from untrusted sources or to use serialization mediums that only permit primitive data types
- **Integrity Checks** - By using a hashing function you can create a digital signature that then can be used later on to verify that the data has not been altered
- **Strict Data Type** - By using strict data types, this can help protect your data from expected data types