Test Website Vulnerabilities Report

TEAM 7.6

SALONI GHULE- 21BCE1967

SHREYA SINGH-21BPS1435

KREET ROUT-21BCE1482

NITIN KUMAR-21BCE1792

Sites Used:

1. http://testfire.net/
2. https://juice-shop.herokuapp.com/#/

1) SQL Injection vulnerability allowing login bypass

CWE-288: Authentication Bypass Using an Alternate Path or Channel

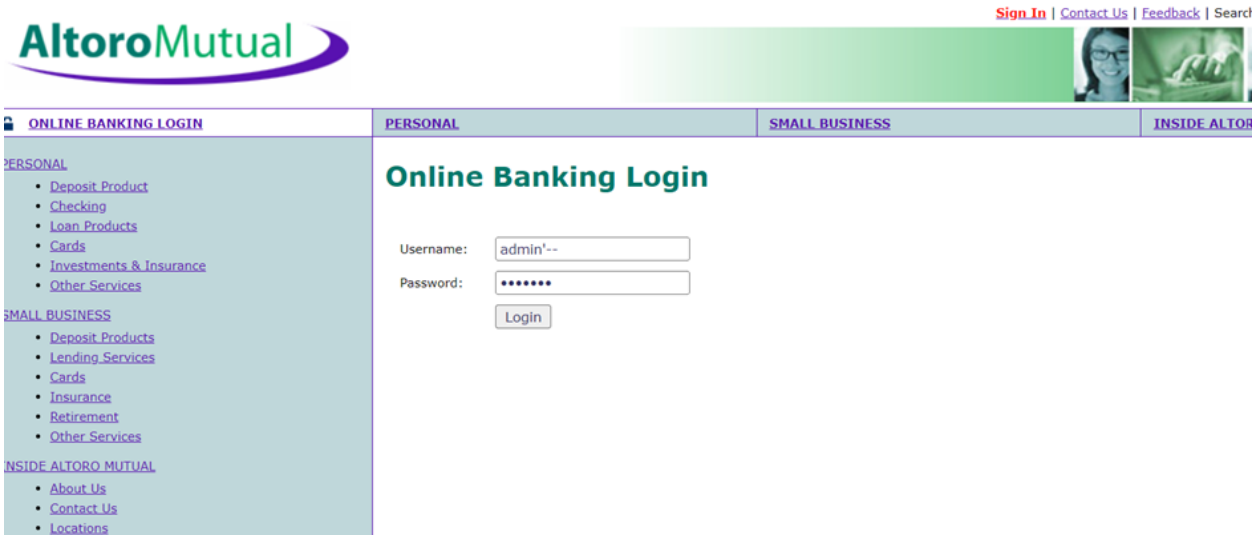
Description: A product requires authentication, but the product has an alternate path or channel that does not require authentication.

Business Impact:

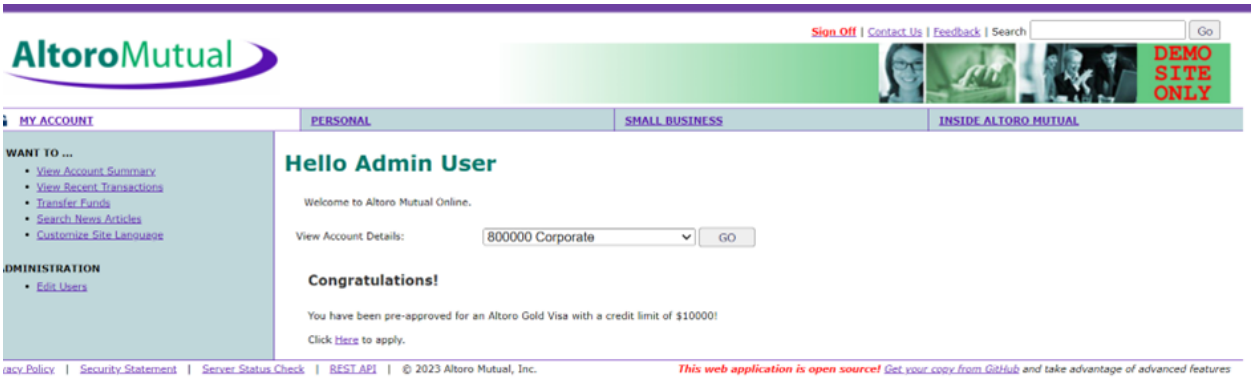
SQL injection attacks represent an extreme security danger to associations. A successful SQL injection assault can bring about confidential and important

information being erased, edited or taken out for malicious uses. Other risks are sites being ruined, defaced or unapproved access to frameworks or accounts and, eventually, compromised machines or whole systems.

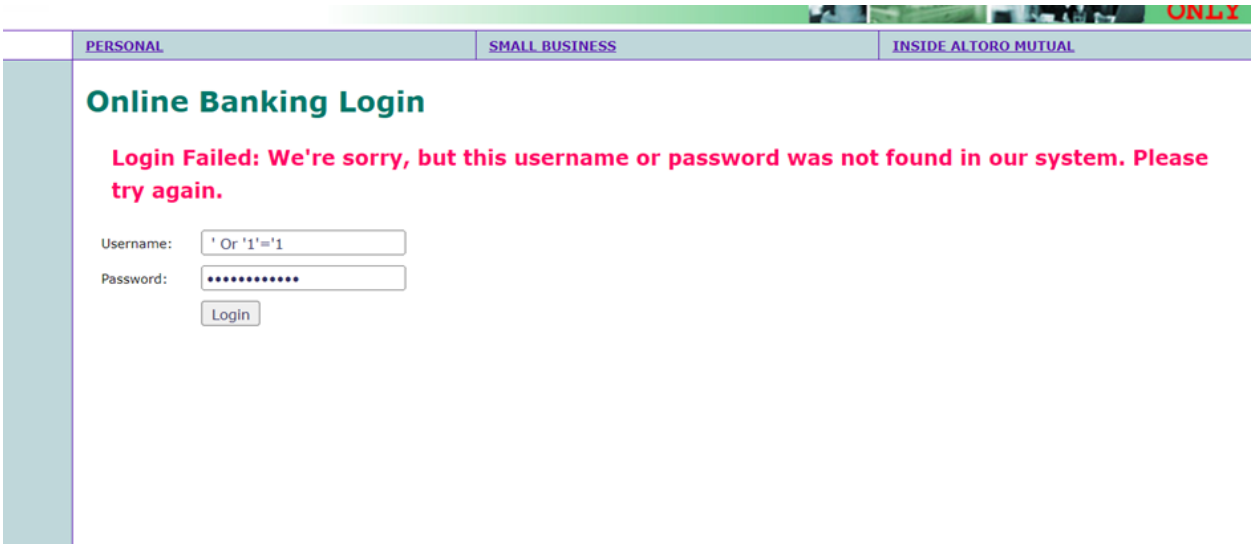
Testing:

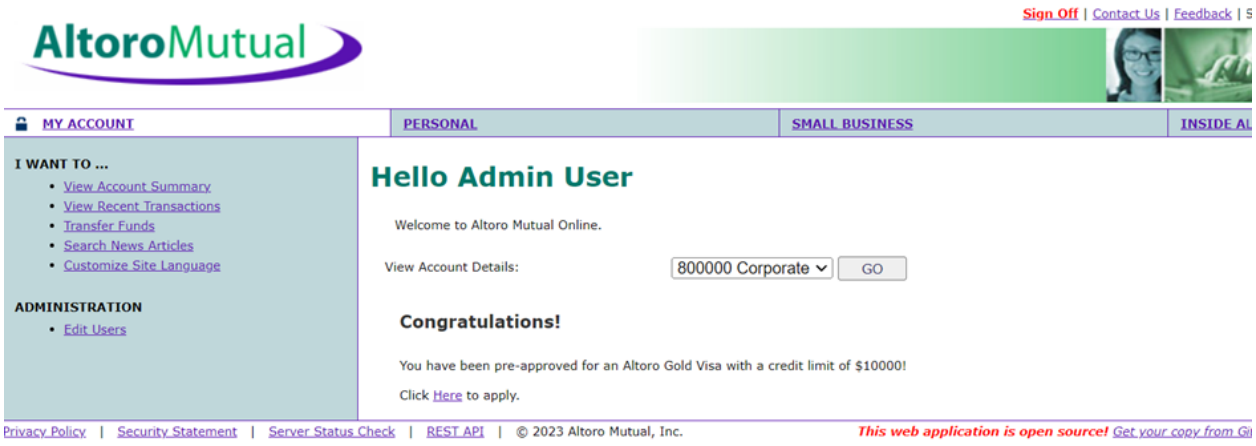


We used the command admin'-- as the username what it signifies is that for the username administrator login into the application without checking the password because -- is used which comments out whatever is written ahead of it thus commenting out the password matching SQL query.



Some other SQL Injections that we can use can be the basic ' Or '1'='1.





2.Brute Force Attack

CWE-1391: Use of Weak Credentials

Description: The product uses weak credentials (such as a default key or hard-coded password) that can be calculated, derived, reused, or guessed by an attacker.

CWE-307: Improper Restriction of Excessive Authentication Attempts

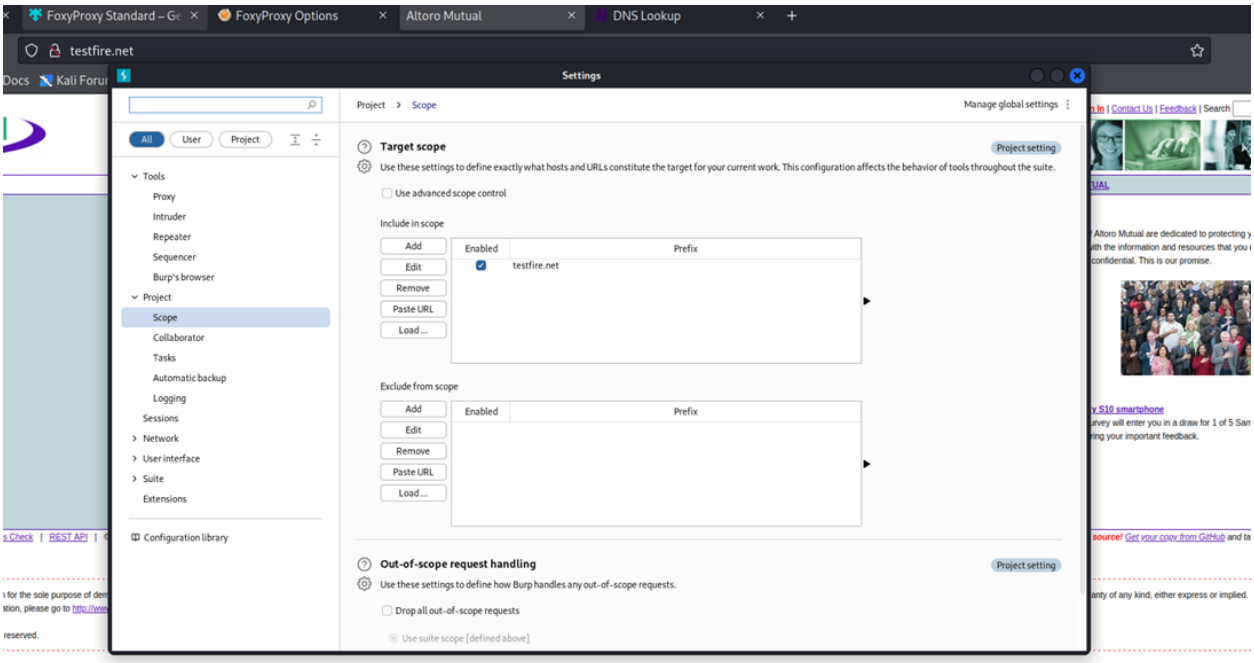
Description: The product does not implement sufficient measures to prevent multiple failed authentication attempts within a short time frame, making it more susceptible to brute force attacks.

Business Impact:

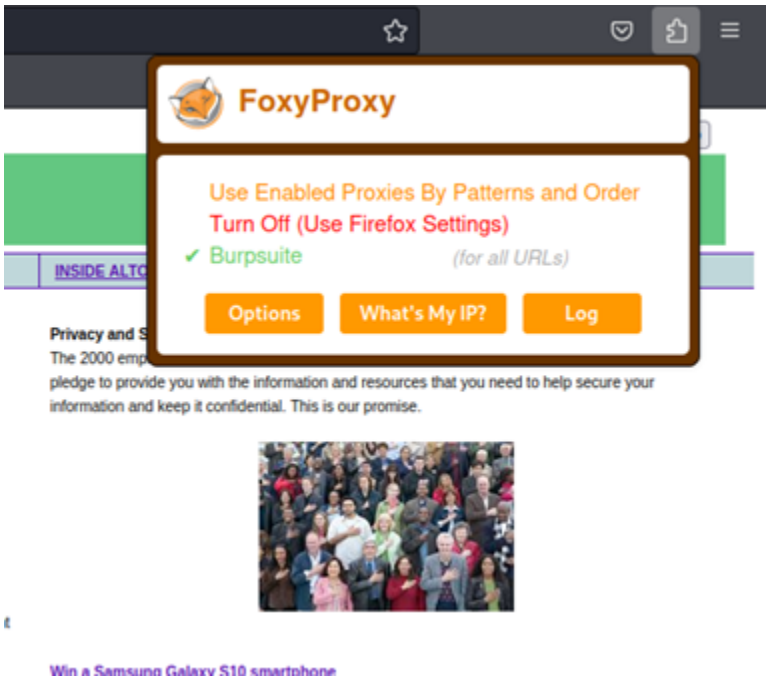
A successful brute force attack on a business can result in unauthorized access, data loss or theft, financial losses, reputational damage, legal consequences, operational disruptions, increased security costs, loss of competitive advantage, damage to trust with customers and partners, and compliance issues, depending on the industry.

Testing:

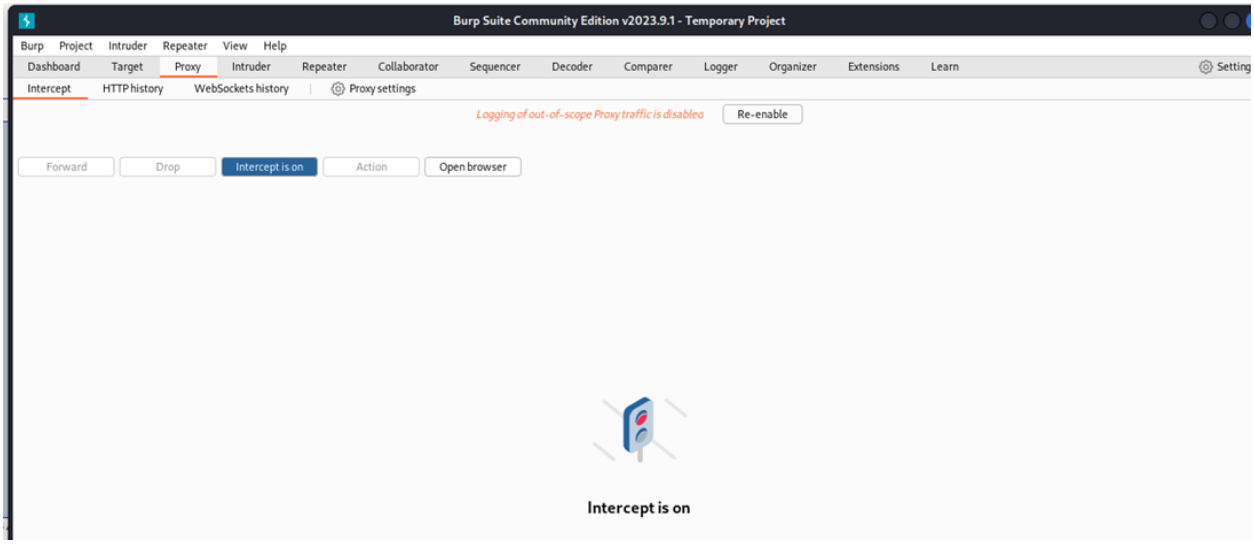
- First we add the website by going to target tab -> add

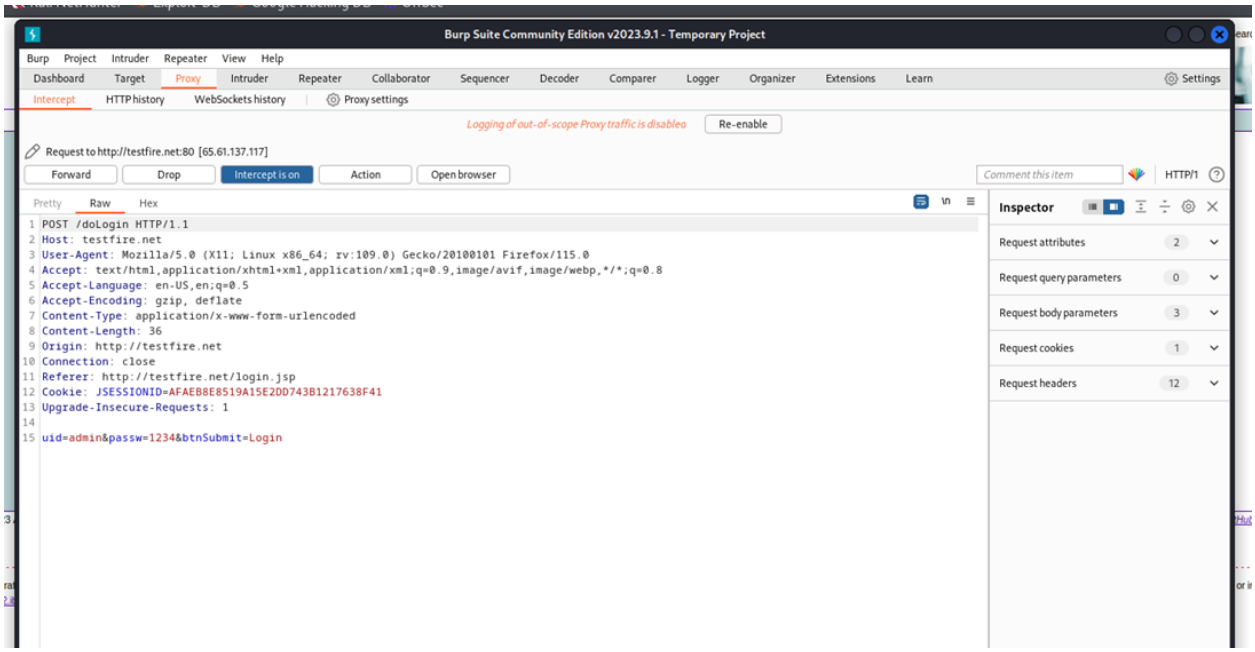


- · Choose burp as our default proxy on foxyproxy.

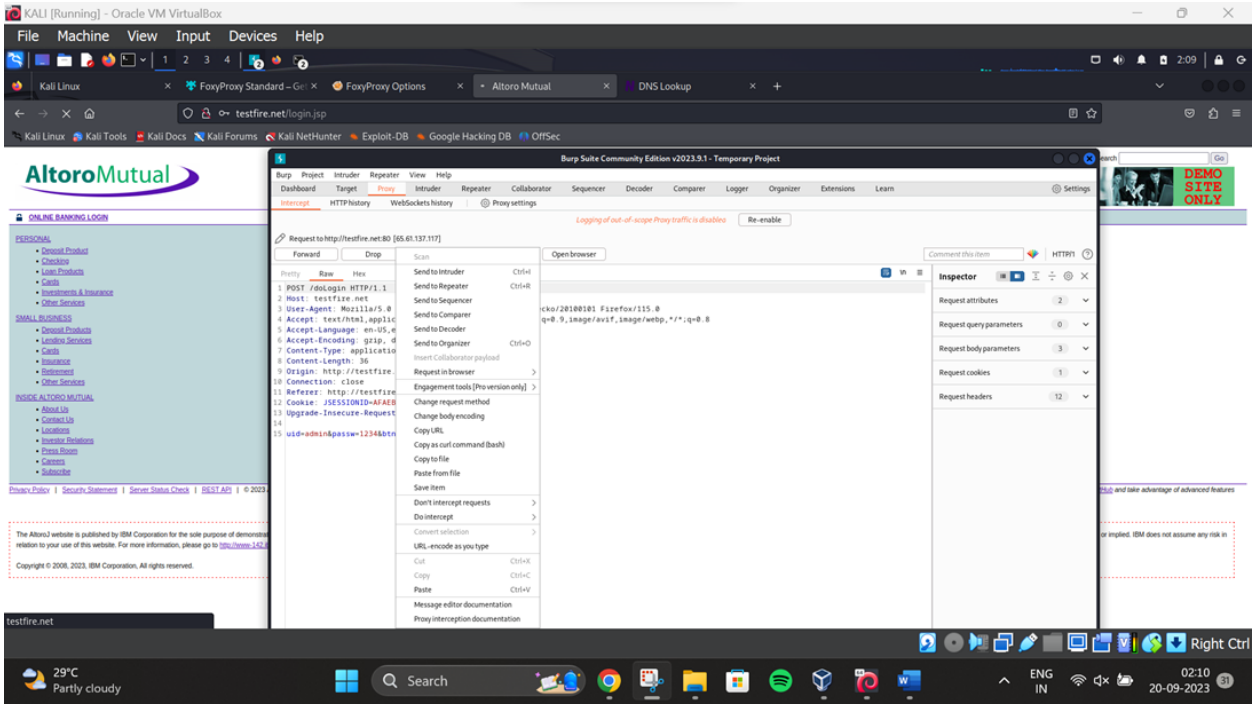


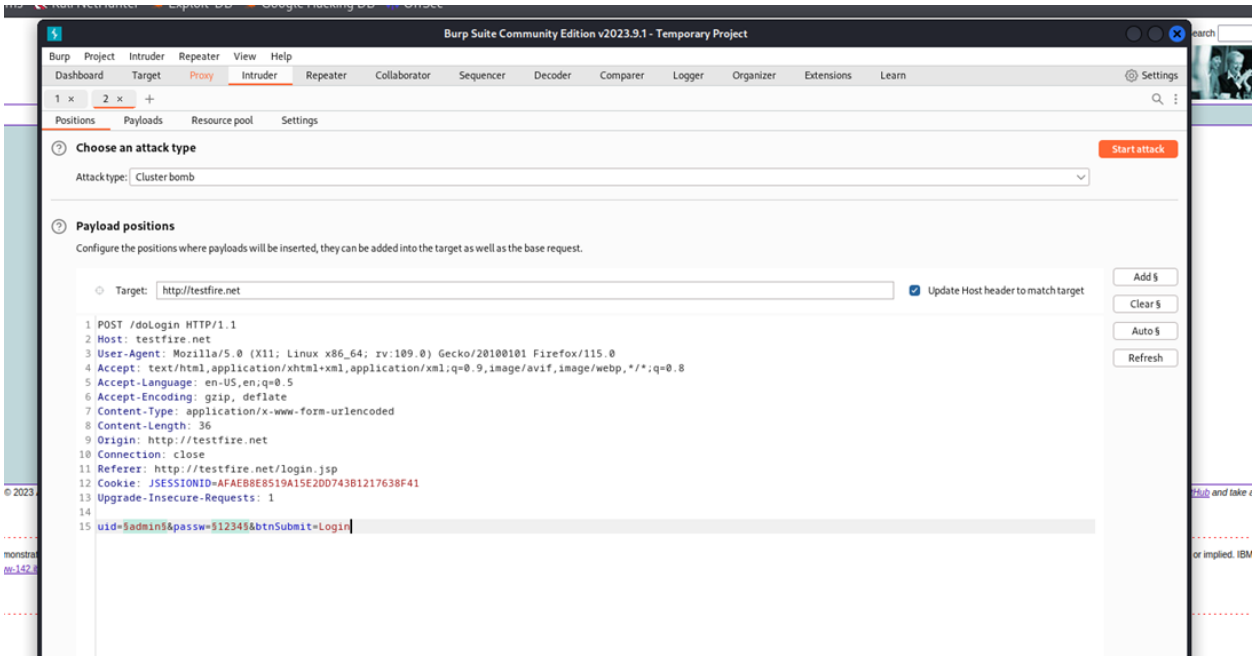
- Go to proxy and turn on the intercept and then click on the login page of the website and give in random username and password.



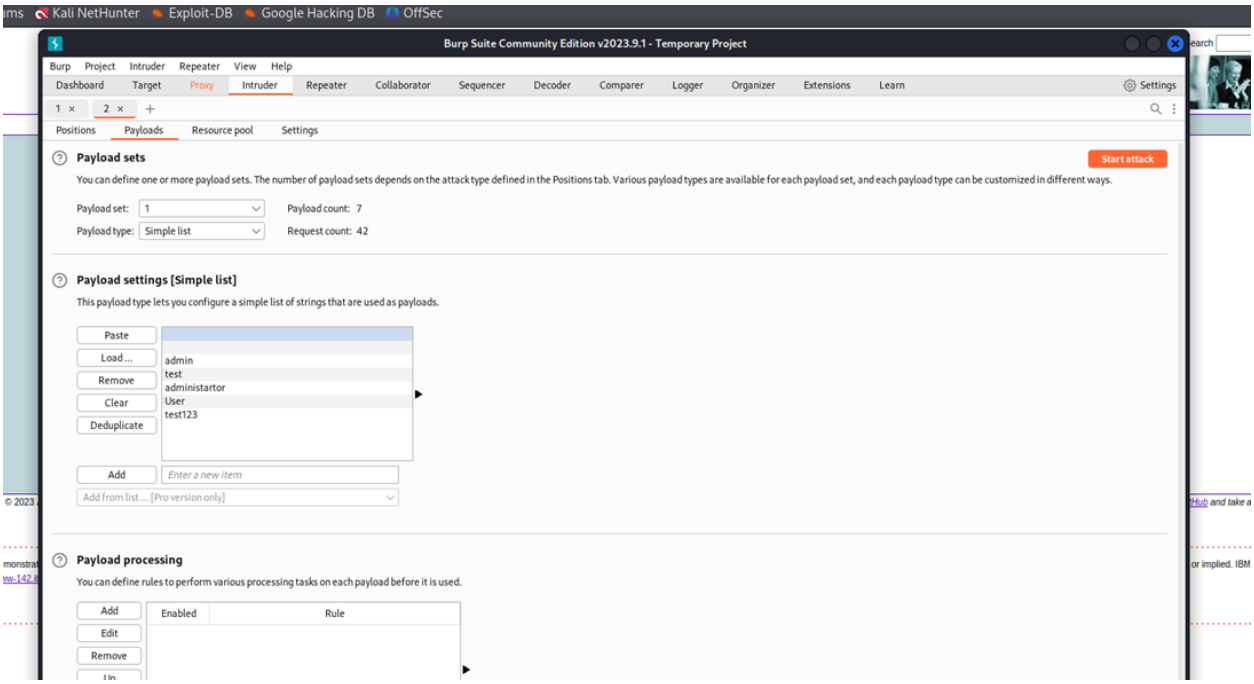


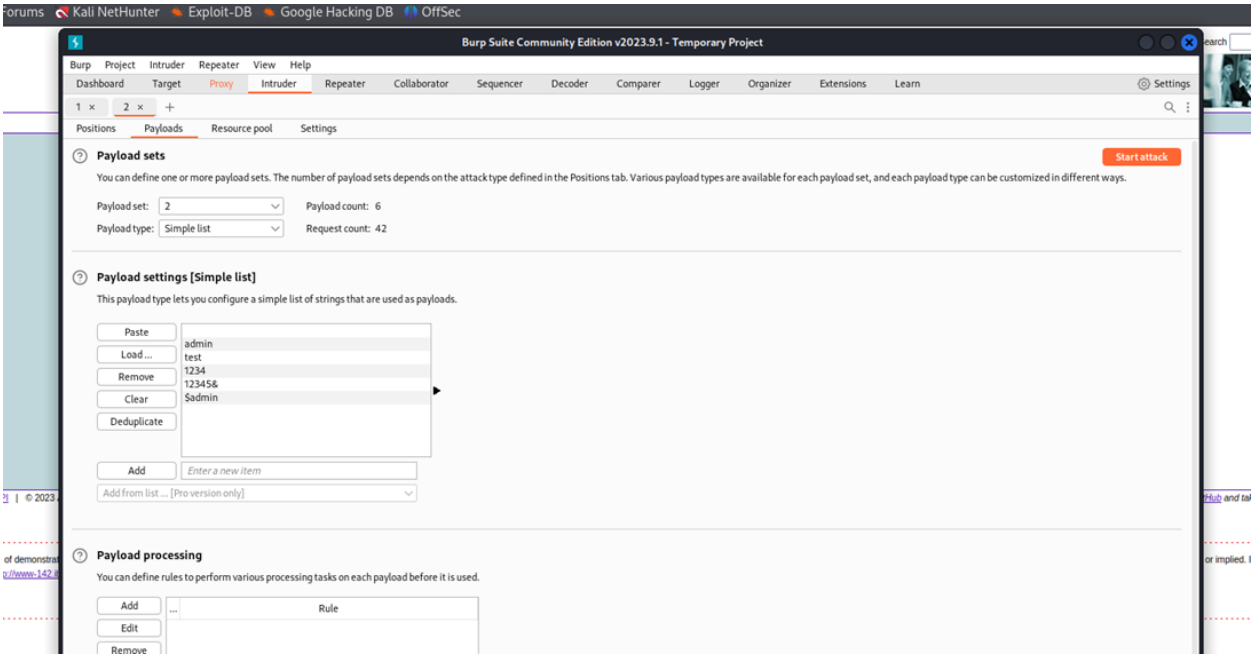
- Then we send it to intruder and go to positions tab choose cluster bomb attack and select the given input of username as payload 1 and given input of password as payload 2 by clicking on add.


KALI [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Kali Linux
FoxyProxy Standard – Ge
FoxyProxy Options
Altoro Mutual
DNS Lookup
testfire.net/login.jsp
Kali Linux Kali Tools Kali Docs Kali Forums Kali NetHunter Exploit-DB Google Hacking DB OffSec
AltoroMutual
ONLINE BANKING LOGIN
PERSONAL
SMALL BUSINESS
INSIDE ALTORO MUTUAL
Burp Suite Community Edition v2023.9.1 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn
Intercept HTTP history WebSockets history Proxy settings
Request to http://testfire.net:80 [65.61.137.117]
Forward Drop
Open browser
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Send to Organizer Ctrl+O
Insert Collaborator payload
Request in browser
Engagement tools [Pro version only]
Change request method
Change body encoding
Copy URL
Copy as curl command (bash)
Copy to file
Paste from file
Save item
Don't intercept requests
Do intercept
Convert selection
URL-encode as you type
Cut Ctrl+X
Copy Ctrl+C
Paste Ctrl+V
Message editor documentation
Proxy interception documentation
Inspector
Request attributes
Request query parameters
Request body parameters
Request cookies
Request headers
DEMO SITE ONLY
testfire.net
Right Ctrl
29°C Partly cloudy
Search
ENG IN
02:10 20-09-2023

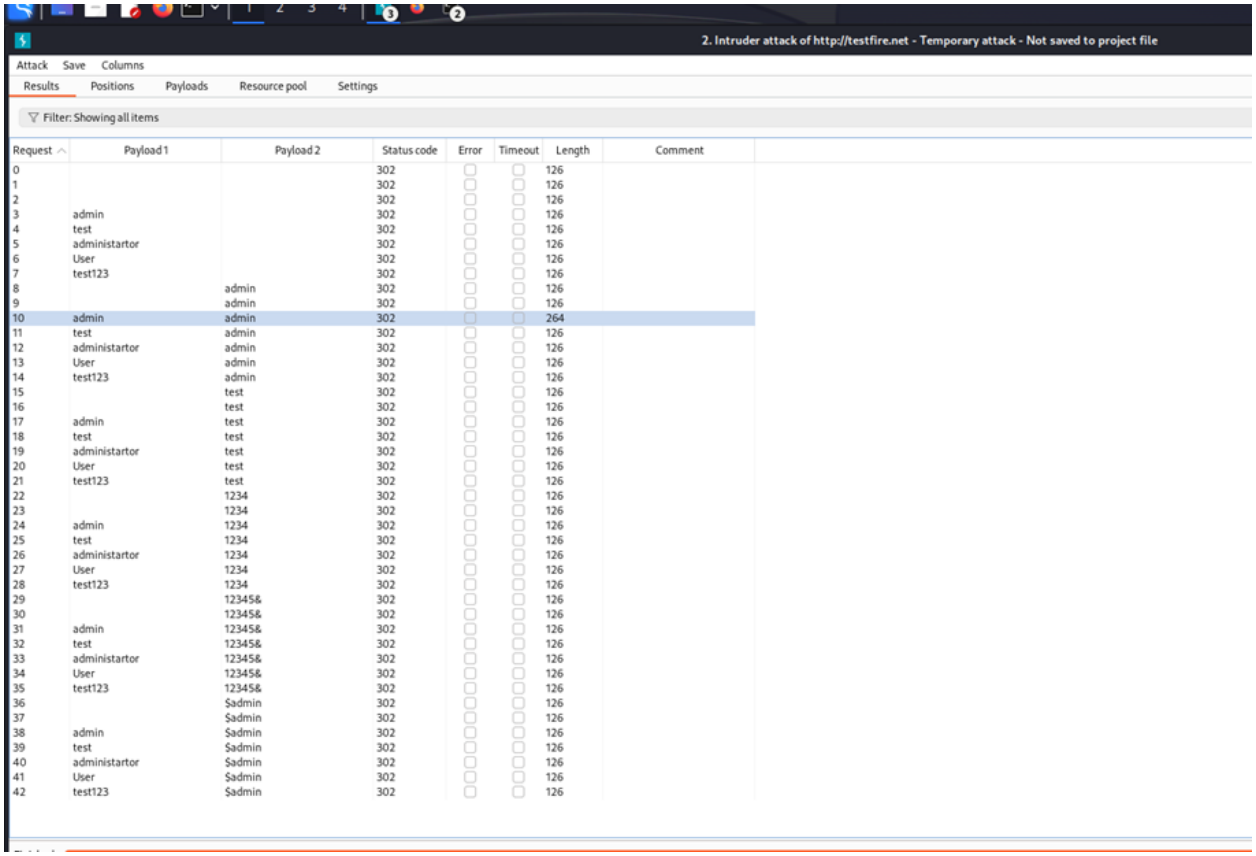

Burp Suite Community Edition v2023.9.1 - Temporary Project
Burp Project Intruder Repeater View Help
Dashboard Target Proxy Intruder Repeater Collaborator Sequencer Decoder Comparer Logger Organizer Extensions Learn Settings
1 × 2 ×
Positions Payloads Resource pool Settings
Choose an attack type
Start attack
Attack type: Cluster bomb
Payload positions
Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.
Target: http://testfire.net
Update Host header to match target
Add §
Clear §
Auto §
Refresh
1 POST /doLogin HTTP/1.1
2 Host: testfire.net
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Content-Type: application/x-www-form-urlencoded
8 Content-Length: 36
9 Origin: http://testfire.net
10 Connection: close
11 Referer: http://testfire.net/login.jsp
12 Cookie: JSESSIONID=AFAEB8E8519A15E2DD743B1217638F41
13 Upgrade-Insecure-Requests: 1
14
15 uid=§admin§&passw=§1234§&btnSubmit=Login

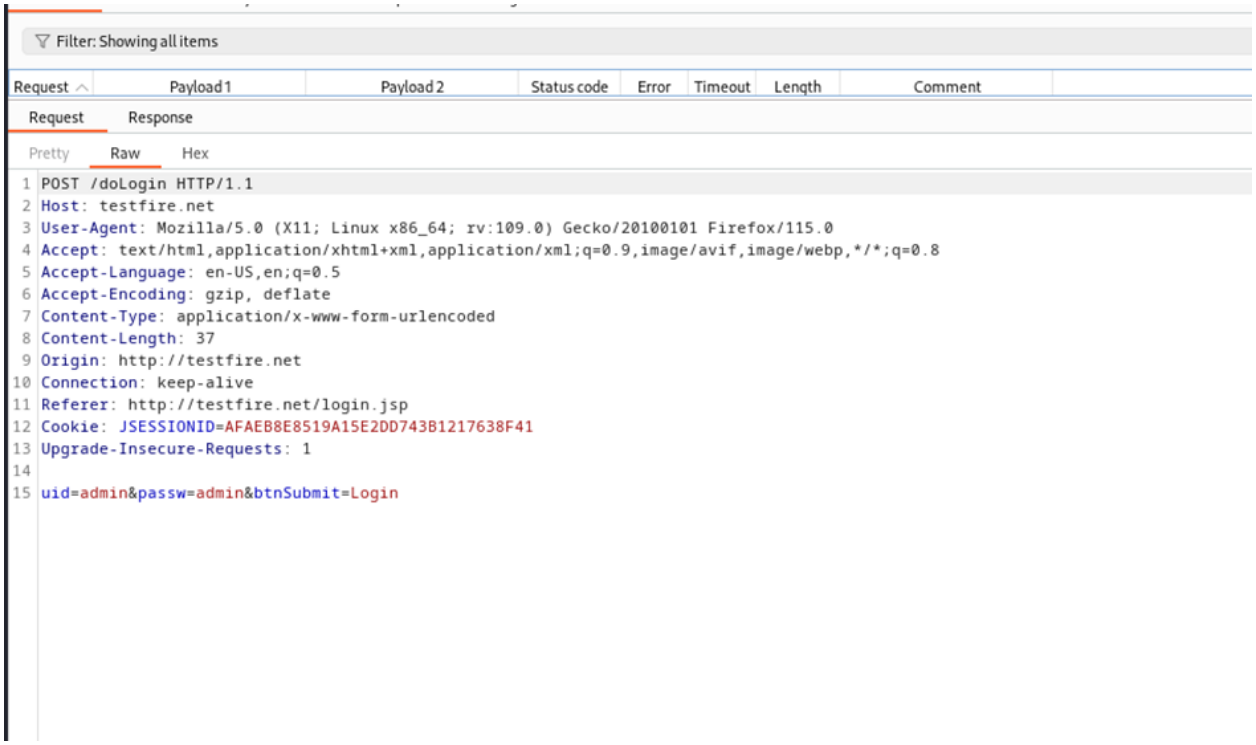
- Then we go to the payloads tab and select payload 1 that is our username in this case and choose simple text and below give some random expected usernames. We can also upload a file here but since I do not have one I did it this way. We do the same for payload 2 which is our passwords and then start the attack.



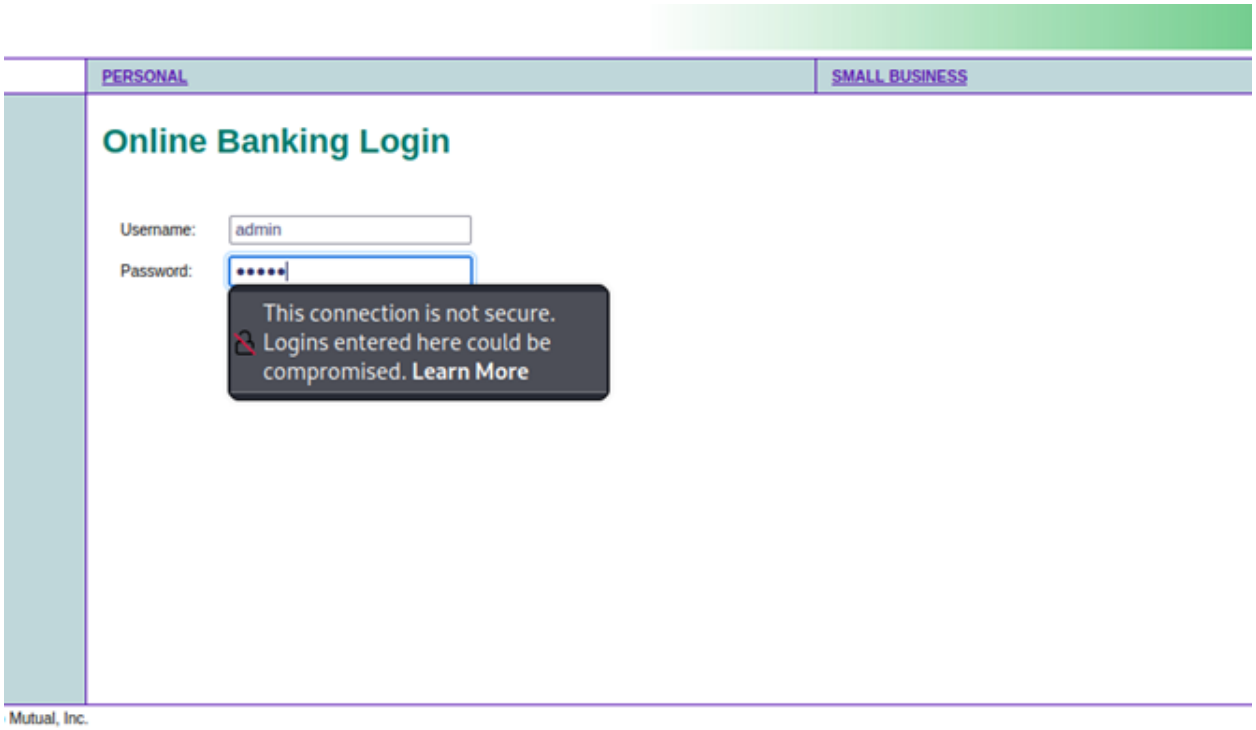


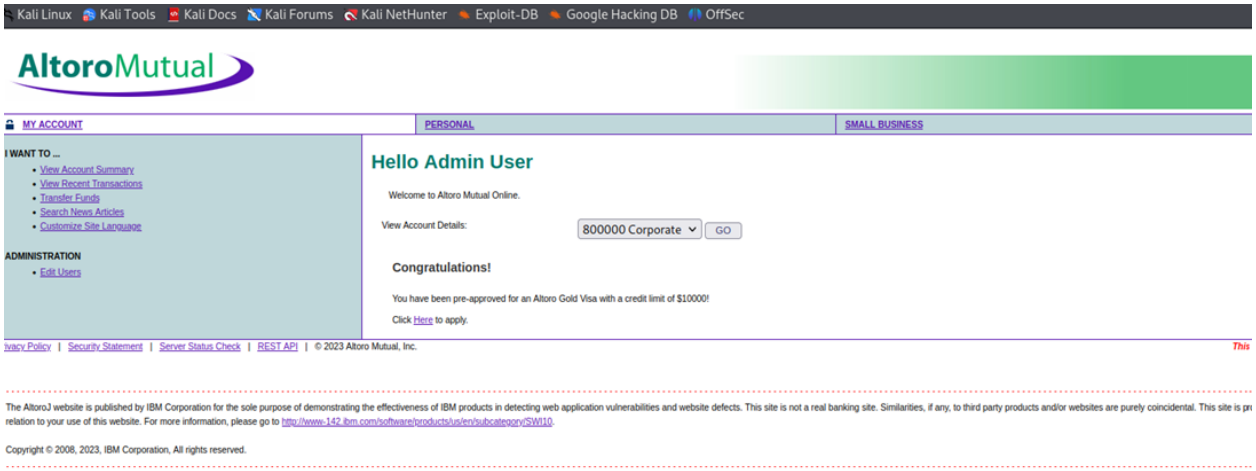
- After the attack is finished we can see that the highlighted admin has different length from the others. Thus it can be a probable solution. Upon checking Request and response we can assure that this is working.





- We then give the inputs in the login page and hence we are logged in.


PERSONAL
SMALL BUSINESS
Online Banking Login
Username:
admin
Password:
This connection is not secure.
Logins entered here could be
compromised. Learn More
Mutual, Inc.


Kali Linux
Kali Tools
Kali Docs
Kali Forums
Kali NetHunter
Exploit-DB
Google Hacking DB
OffSec
AltoroMutual
MY ACCOUNT
PERSONAL
SMALL BUSINESS
I WANT TO ...
View Account Summary
View Recent Transactions
Transfer Funds
Search News Articles
Customize Site Language
ADMINISTRATION
Edit Users
Hello Admin User
Welcome to Altoro Mutual Online.
View Account Details:
800000 Corporate
GO
Congratulations!
You have been pre-approved for an Altoro Gold Visa with a credit limit of $10000!
Click Here to apply.
ivacy Policy | Security Statement | Server Status Check | REST API | © 2023 Altoro Mutual, Inc.
This
The AltoroJ website is published by IBM Corporation for the sole purpose of demonstrating the effectiveness of IBM products in detecting web application vulnerabilities and website defects. This site is not a real banking site. Similarities, if any, to third party products and/or websites are purely coincidental. This site is pr
relation to your use of this website. For more information, please go to http://www-142.ibm.com/software/products/us/en/subcategory/SW110.
Copyright © 2008, 2023, IBM Corporation, All rights reserved.

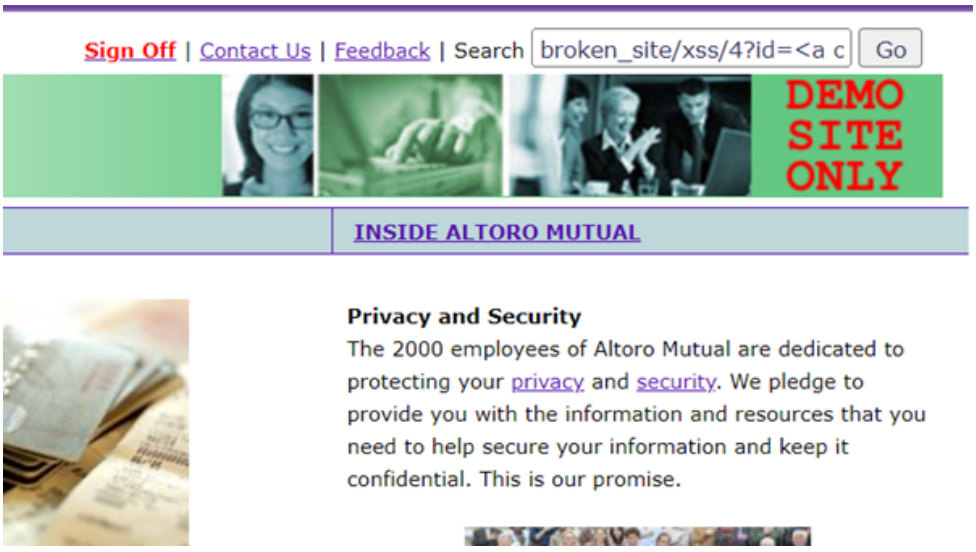
3. Cross Site Scripting (XSS)

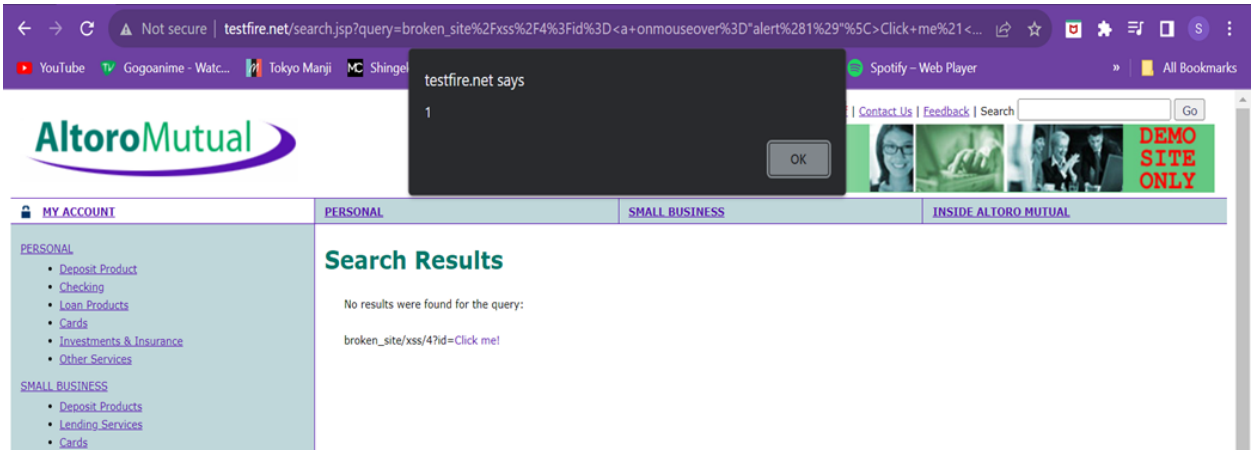
CWE-87: Improper Neutralization of Alternate XSS Syntax

Description: The product does not neutralize or incorrectly neutralizes user-controlled input for alternate script syntax.
Business Impact: Cross-Site Scripting (XSS) attacks in business can lead to data theft, user trust erosion, reputation damage, financial losses, legal consequences, operational disruptions, increased security costs, loss of competitive advantage, and customer trust issues.

Testing:

Injecting a basic a tag and seeing how the site reacts to that. Payload is **broken_site/xss/4?id=Click me!**. What we do here is basically create a link that says click me, and when we hover the mouse over it, the alert box should execute.





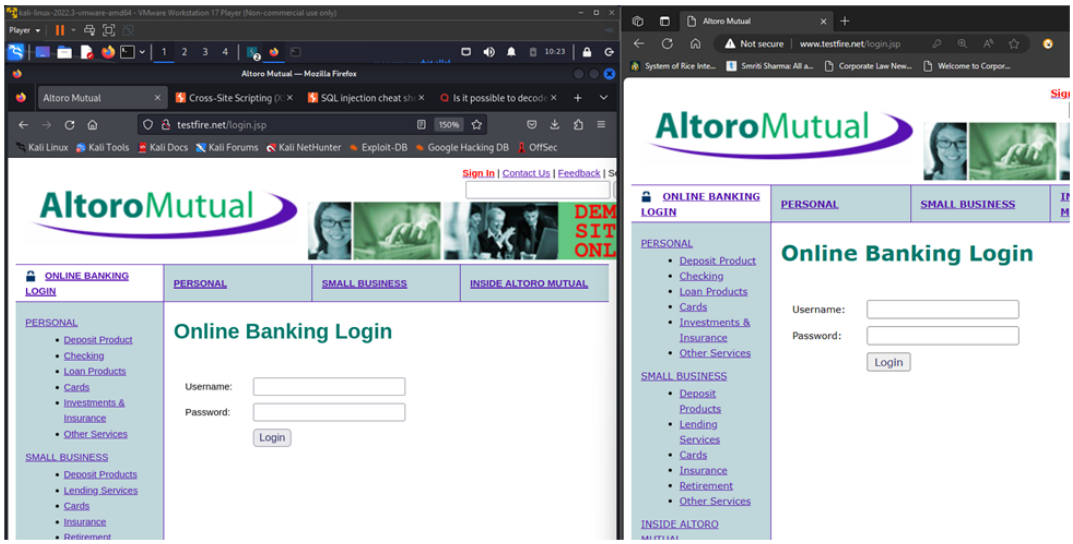
4. Improper Session Management

CWE ID for Session Fixation: CWE-384 (Session Fixation)

Description: Attacker sets or fixes a user's session ID, leading to unauthorized access. This can occur when session management is improperly implemented or lacks adequate security controls.

Business Impact: The business impact of CWE-384 (Session Fixation) can be significant, potentially leading to unauthorized access to user accounts. This can result in data breaches, loss of sensitive information, reputation damage, and financial losses due to legal and remediation costs.

Testing:

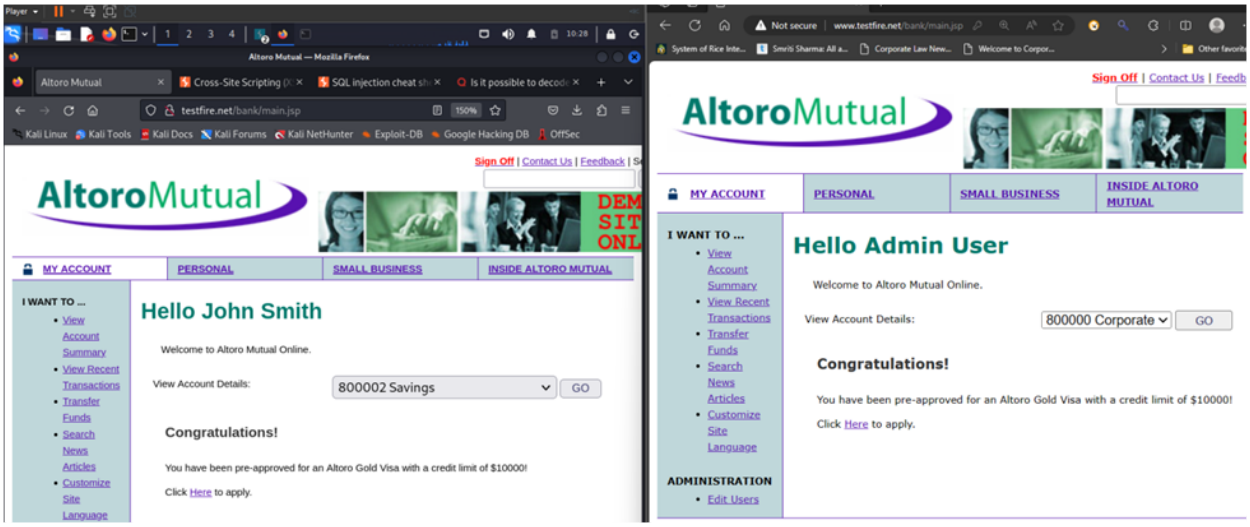


We have accessed the testfire.net website on two different browsers(one on virtual machine and another on main device).

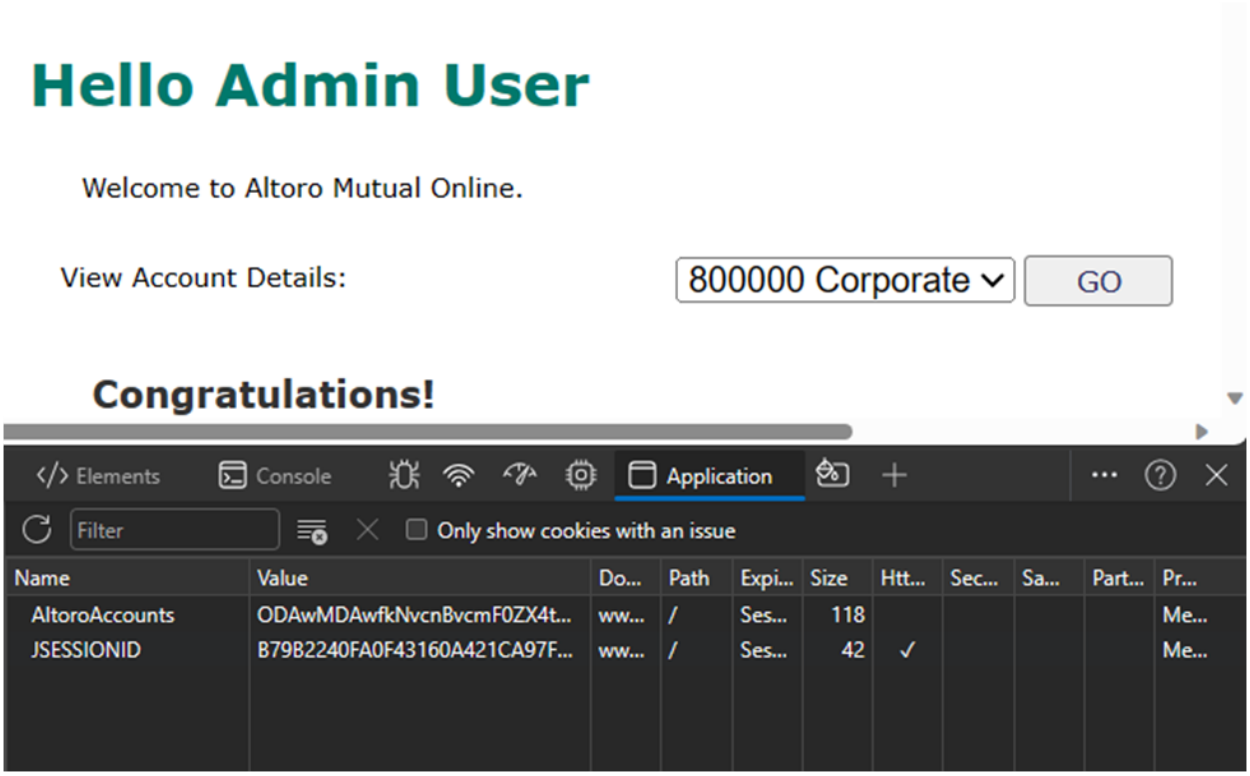
We will login as admin (u_name = admin && pass = admin) in the main machine and as John Smith (u_name = jsmith && pass = Demo1234) in the virtual machine.

Post login, we will fetch the JSESSION_ID of admin logged in account and copy that, intercept a request in the John Smith's account and replace John Smith's Session Id with admin's.

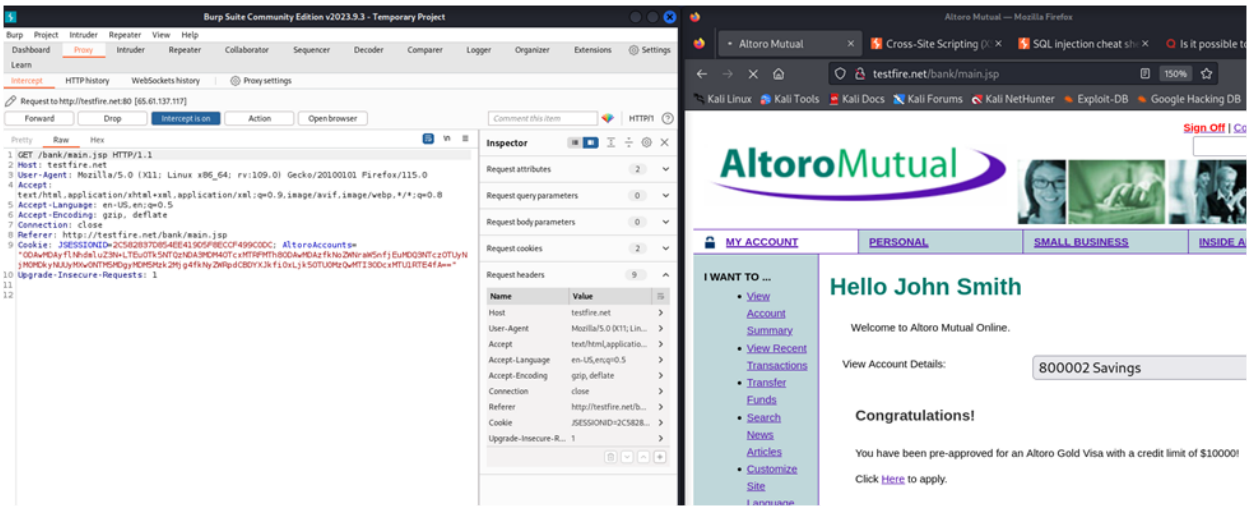
If we are able to access the admins account after this then we could possibly say that theres a security flaw of insecure Session management.



- Logged in to respective accounts.



- Fetched the Admin Session ID value.(B79B2240FA0F43160A421CA97F1343C6)



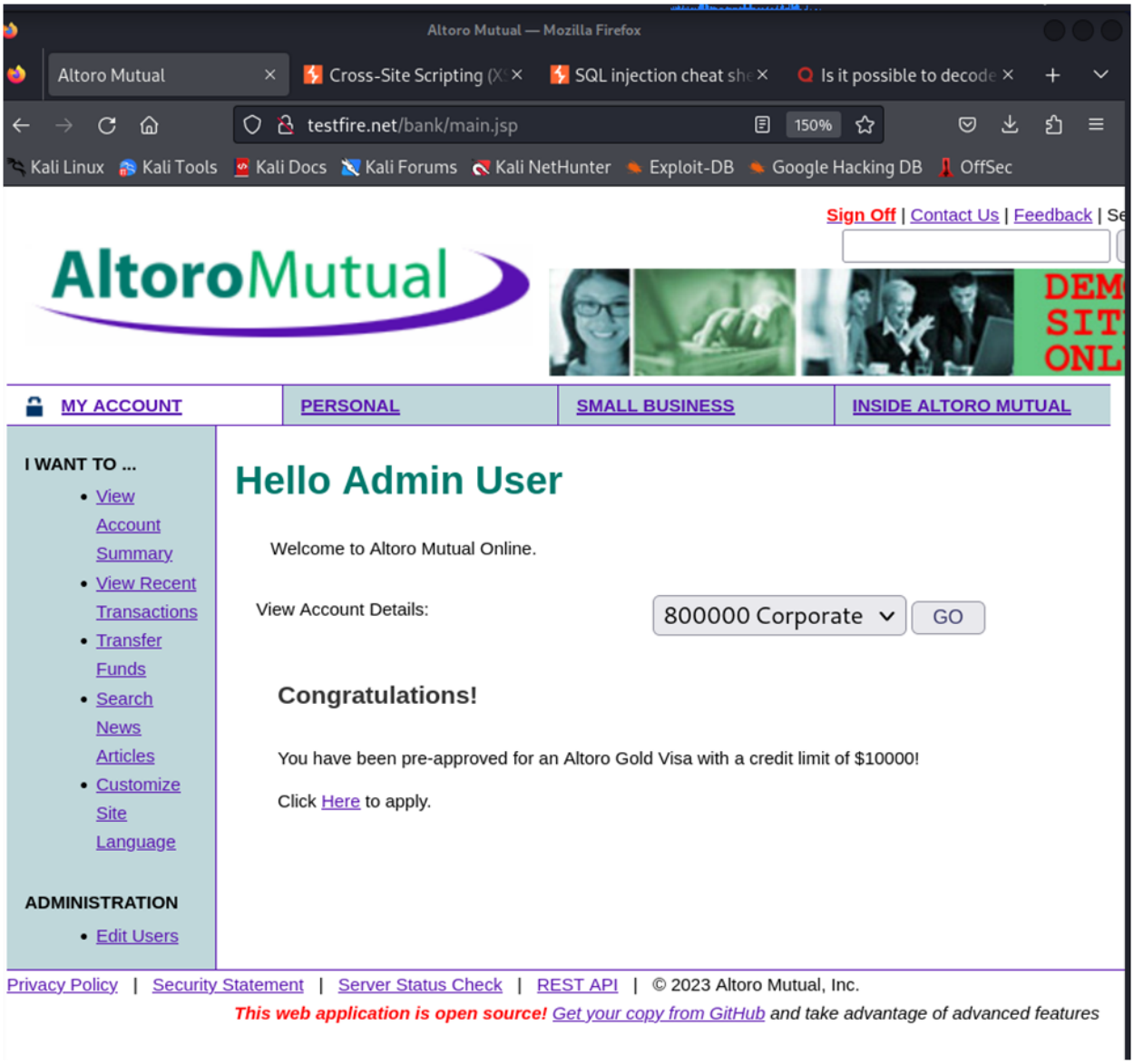
- Intercepted a request from John Smith’s account.

```
1  GET /bank/main.jsp HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Referer: http://testfire.net/bank/main.jsp
9  Cookie: JSESSIONID=2C582837D854EE419D5F8ECCF499C0DC; AltoroAccounts=
   "ODAwMDAyflNhdmluZ3N+LTEuOTk5NTQzNDA3MDM4OTcxMTRFMTh8ODAwMDAzfkNoZWNraW5nfjEuMDQ3NTczOTUyN
   jM0MDkyNUUyMXw0NTM5MDgyMDM5Mzk2Mjg4fkNyZWRpdCBDYXJkfi0xLjk5OTU0MzQwMTI3ODcxMTUlRTE4fA=="
10 Upgrade-Insecure-Requests: 1
11
12
```

- The intercepted request has JSESSIONID value as 2C582837D854EE419D5F8ECCF499C0DC

Pretty Raw Hex \n

```
1  GET /bank/main.jsp HTTP/1.1
2  Host: testfire.net
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:109.0) Gecko/20100101 Firefox/115.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Referer: http://testfire.net/bank/main.jsp
9  Cookie: JSESSIONID=B79B2240FA0F43160A421CA97F1343C6; AltoroAccounts=
   "ODAwMDAyflNhdmluZ3N+LTEuOTk5NTQzNDA3MDM4OTcxMTRFMTh8ODAwMDAzfkNoZWNraW5nfjEuMDQ3NTczOTUyN
   jM0MDkyNUUyMXw0NTM5MDgyMDM5Mzk2Mjg4fkNyZWRpdCBDYXJkfi0xLjk5OTU0MzQwMTI3ODcxMTUlRTE4fA=="
10 Upgrade-Insecure-Requests: 1
11
12
```



- Replaced the JSESSIONID with that of admin's

Logged in as admin

5.Broken Access Control:

CWE-284: Improper Access Control.

Description: The product does not restrict or incorrectly restricts access to a resource from an unauthorized actor.

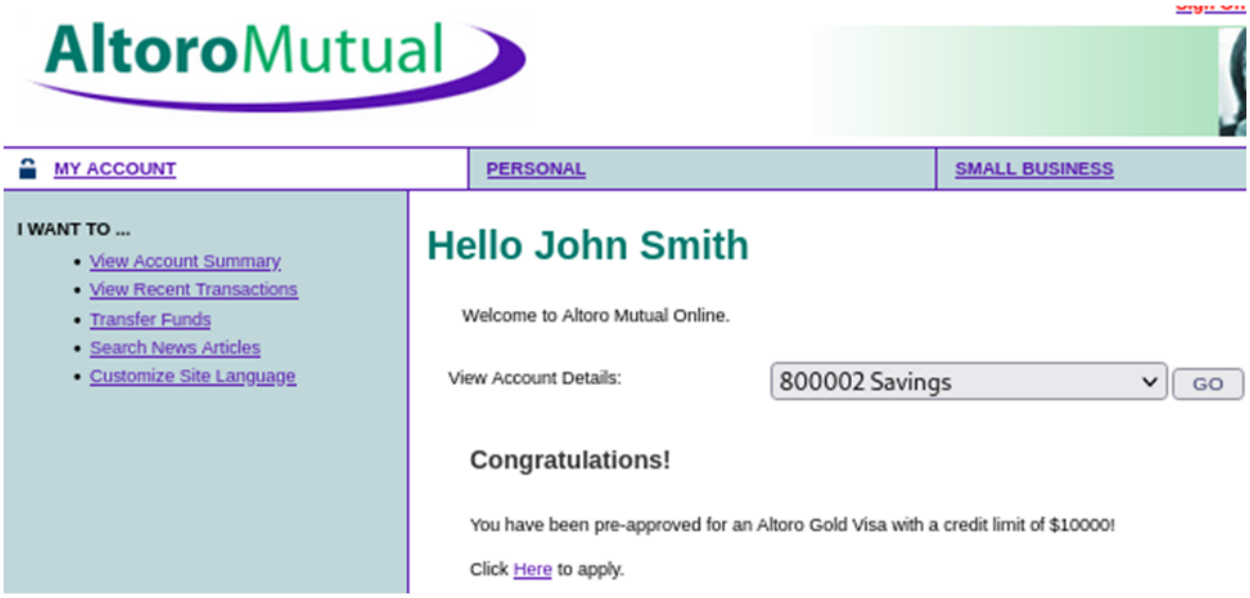
Business Impact:

Data Breaches, Loss of Confidential Information, Financial Loss, Reputation Damage, Regulatory Compliance Issues, Operational Disruption, Lawsuits and Legal Liabilities, Resource Wastage, Loss of Competitive Advantage, Customer Trust Erosion

Testing:

We will first login as john smith and then try to access a bank account linked to any other user to view their transaction by modifying the url parameter (IDOR attack).

- Logged in as John Smith

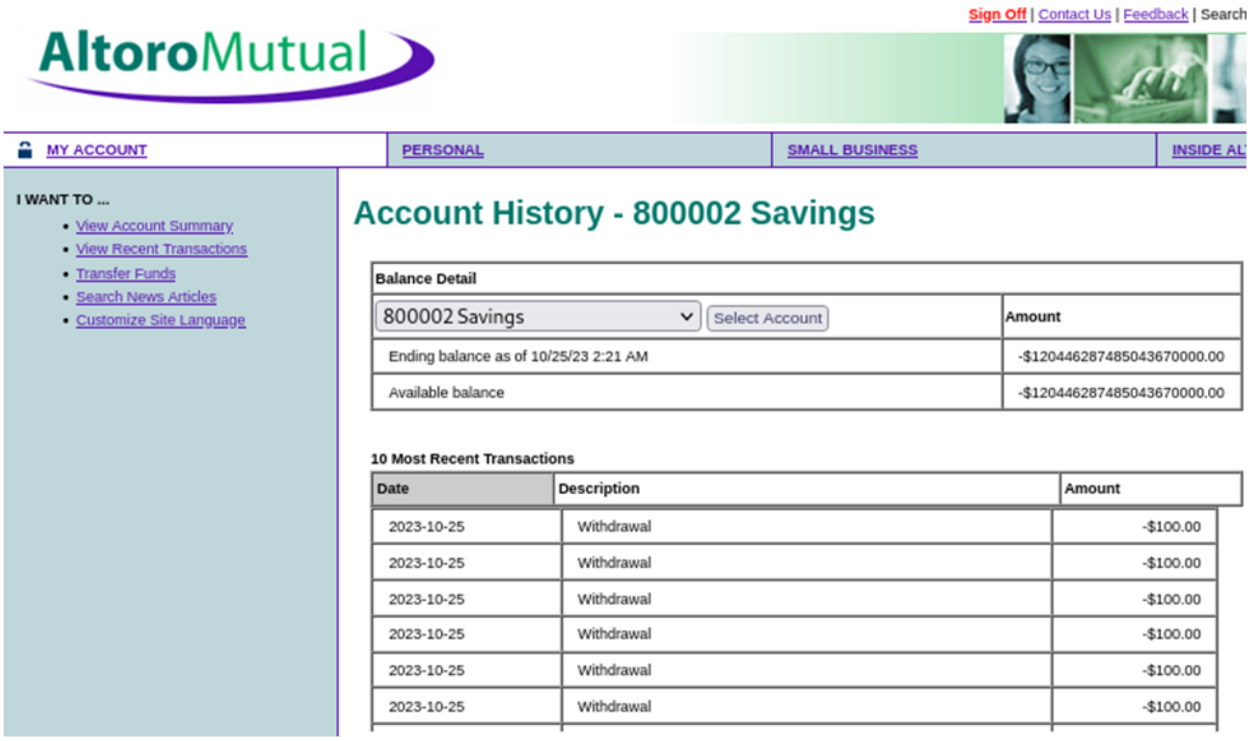


Here if we go to the view account summary section, we will notice an option to search the account history of all the accounts we posses. John Smith has access to the following accounts:

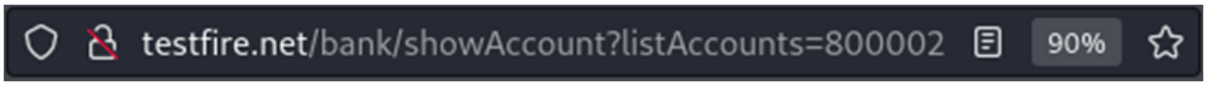
- 800002 – Savings
- 800003 – Checking
- 4539082039396288 – Credit card

If we access his Savings account, we notice account history where the 10 most recent transactions are of $100 each.

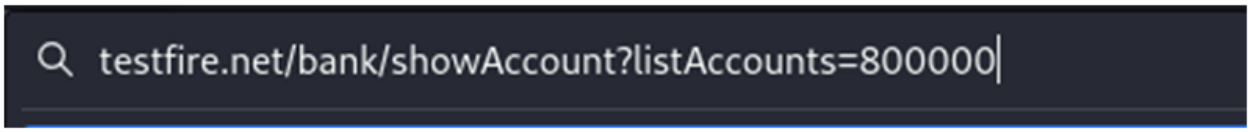
Having a look at the URL we can see the request parameter to be as listAccount=800002.



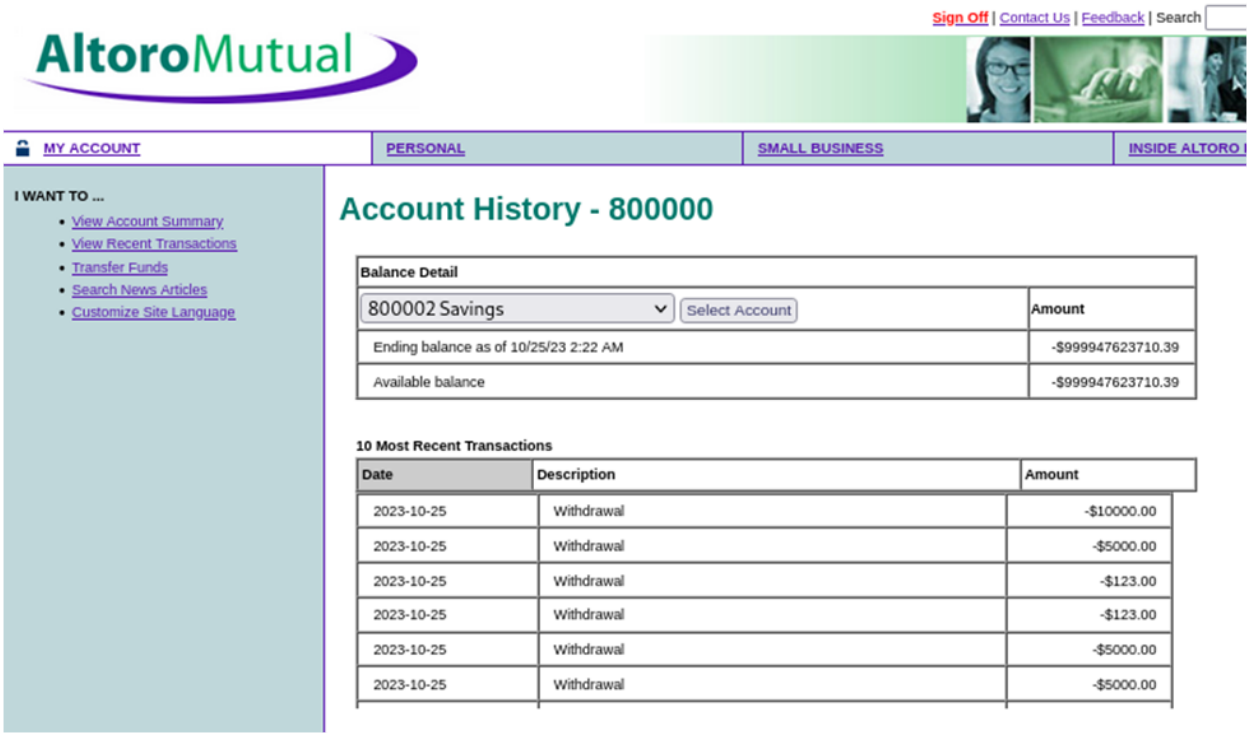
John Smith's Savings account's URL -



From here if we modify the URL to the one shown in the image below with account 800000 which technically doesn't belong to john smith (800000 Belongs to the admin user), we shouldn't be able to access it.



If we send this url request to the server we get a 200 OK response and apparently we are able to view the transaction history of account number 800000 as all the recent transaction values have changed.



Hence we have successfully exploited the IDOR (InDirect Object Reference) vulnerability.

6. HTML injection attack + ClickJacking

CWE=601: URL Redirection to untrusted site('Open Redirect')

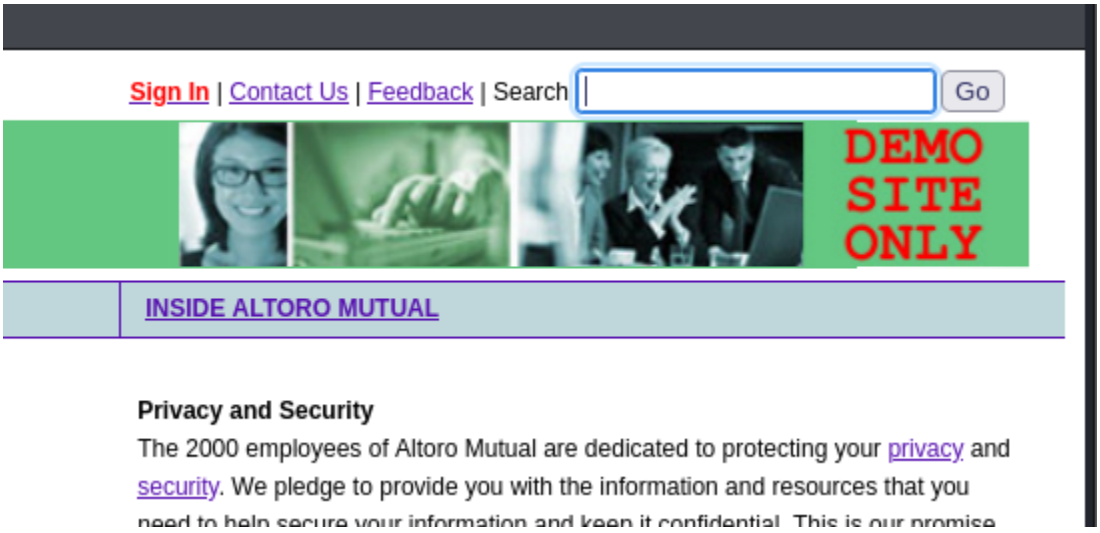
Description: A web application accepts a user-controlled input that specifies a link to an external site, and uses that link in a Redirect. This simplifies phishing attacks.

Business Impact:

The impact of CWE-601 (Open Redirect) includes loss of trust, data breaches, financial loss, legal consequences, brand damage, and operational disruption.

Here a combination of man in the middle attack and html injection can be used to inject an html payload that can return a link to a malicious copy of the login page of the legitimate website seeking the credentials from the user.

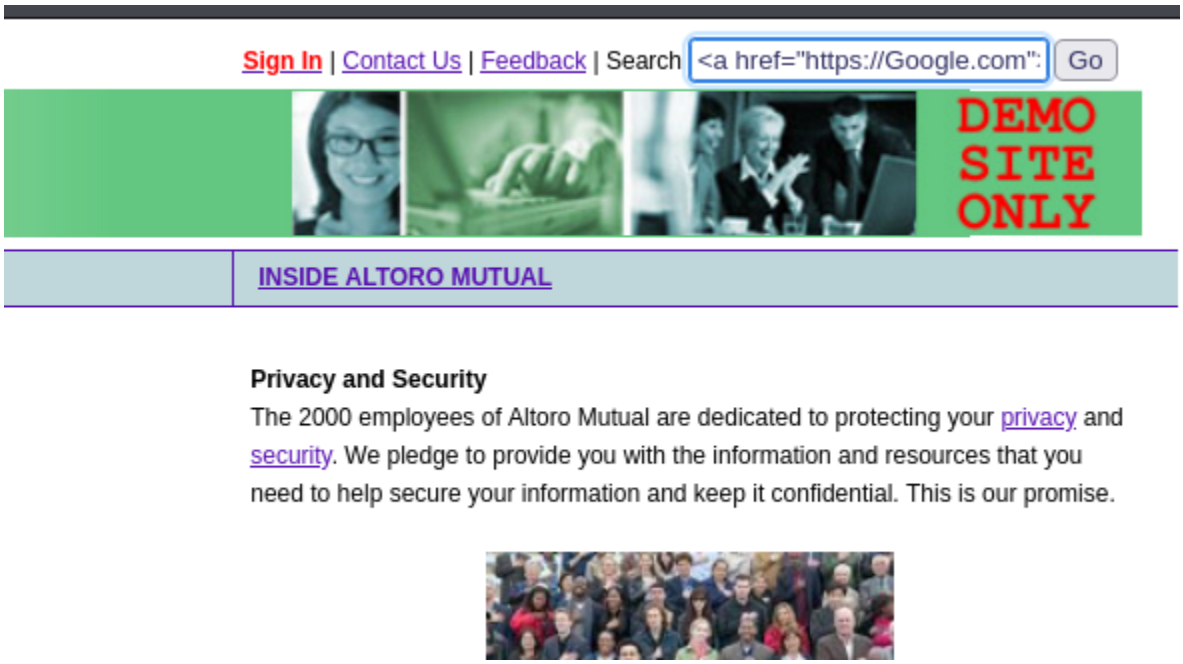
The search bar is vulnerable to html injection



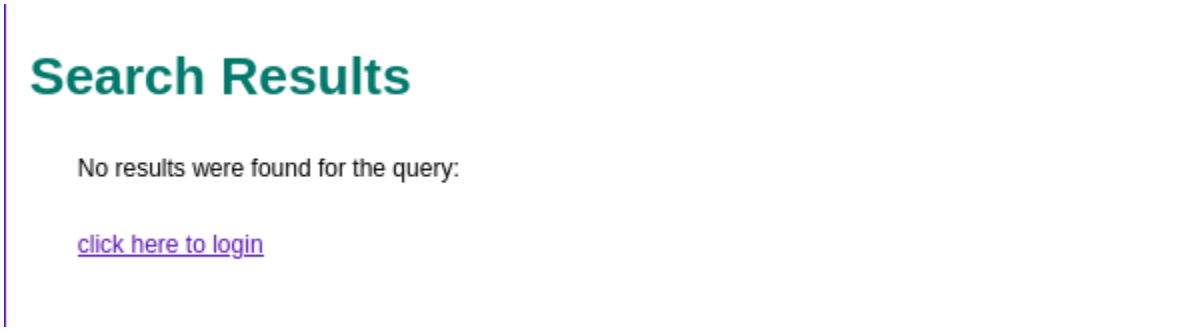
HTML payload:

click here to login

This payload's href link can be modified in certain way that it redirects to the malicious login page.



Executing the payload gives the following result.



When the click here to login hyperlink is clicked, we are redirected to the website that is linked to it.

7. Improper Input Validation

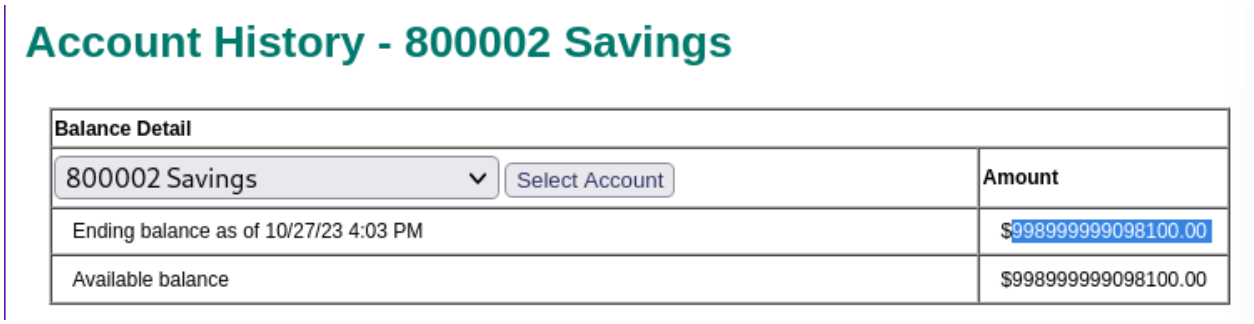
The website allows user to transfer amount greater than the user has in their account.

CWE (Common Weakness Enumeration): CWE-132

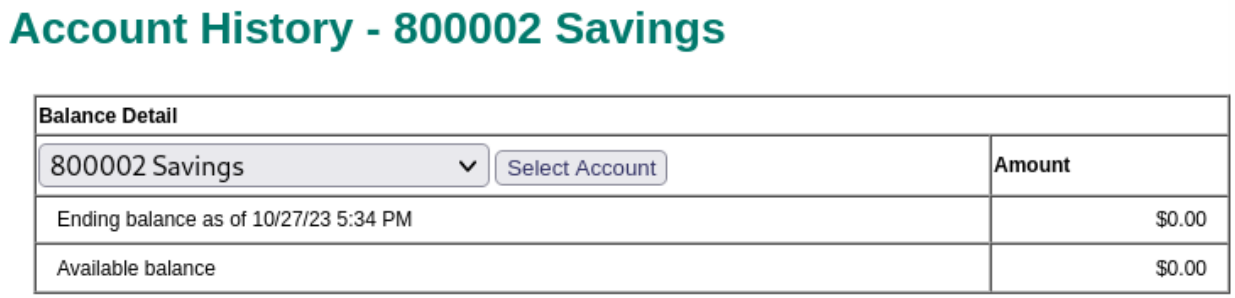
Business Impact:

- Financial losses: Unauthorized transfers could result in substantial financial losses for both the business and affected users.
- Reputation damage: Such a security flaw can erode user trust and damage the reputation of the company.
- Legal and regulatory consequences: Violating financial regulations can result in fines and legal actions against the organization.
- Customer churn: Users may leave the platform due to concerns about their financial security.

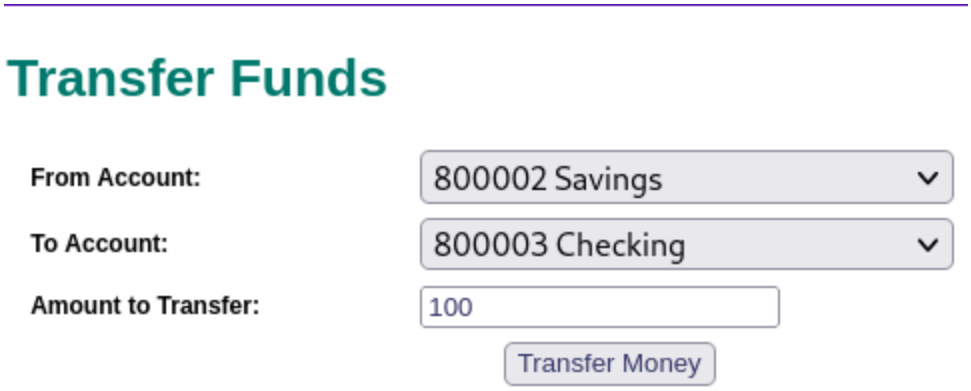
Wearing off the account 800002.

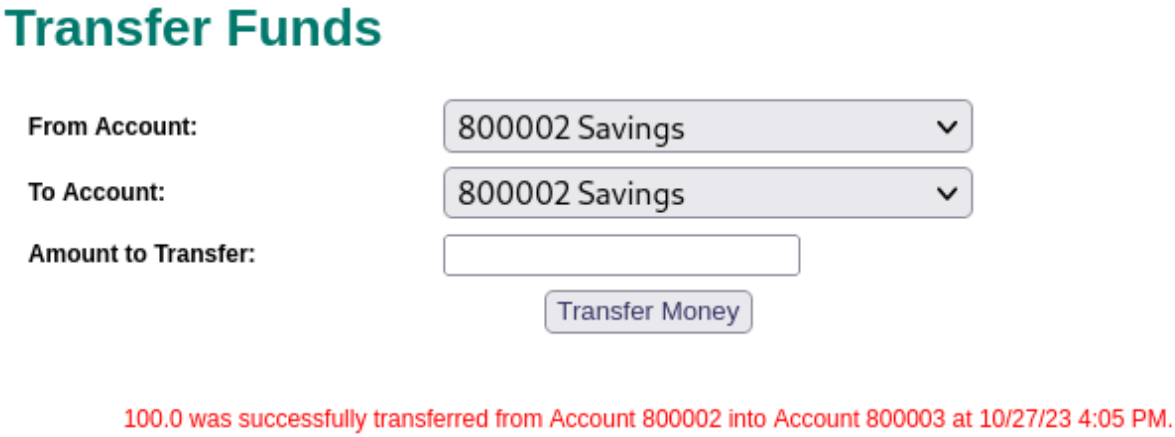


Amount to be debited to make the account balance empty is 998999999098100



Now trying to transfer amount even after the balance has worn out.





The transfer is successful as shown and the amount is credited in the respective account.

8) CRYPTOGRAPHIC FAILURE / SENSITIVE DATA EXPOSURE

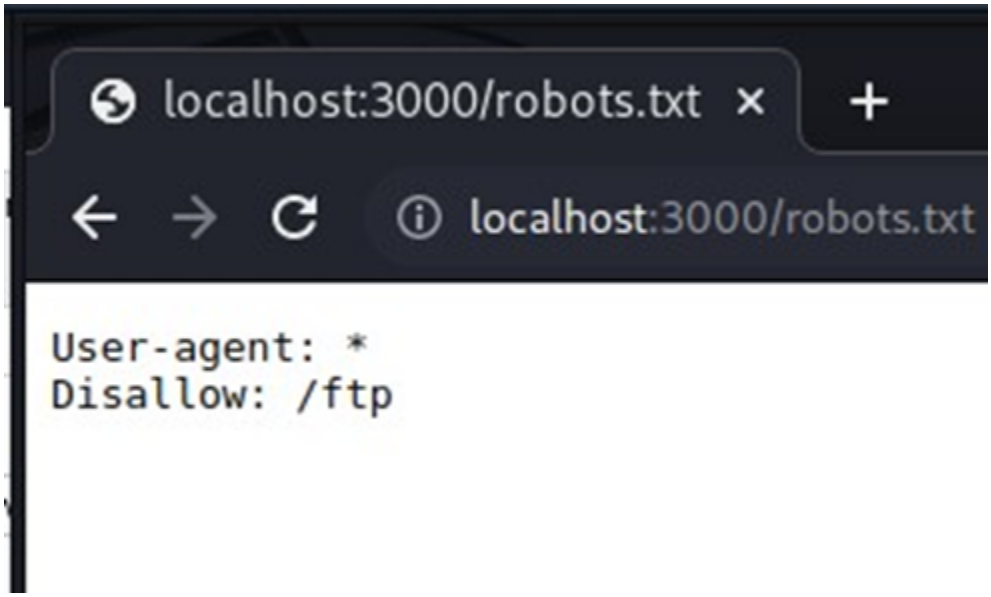
CWE-200: Exposure of Sensitive Information to an Unauthorized Actor
Description: The product exposes sensitive information to an actor that is not explicitly authorized to have access to that information.

Business Impact: Cryptographic failure and the subsequent exposure of sensitive data can have profound business impacts. Financially, it may result in the costs of breach investigations, data recovery, and potential fines, causing significant economic strain. The tarnishing of the company's reputation can lead to the loss of customer trust, hindering revenue generation and making it challenging to acquire new clients. Data theft or exposure can result in identity theft, fraud, and legal consequences, further compounding the overall damage to the business.

Testing:

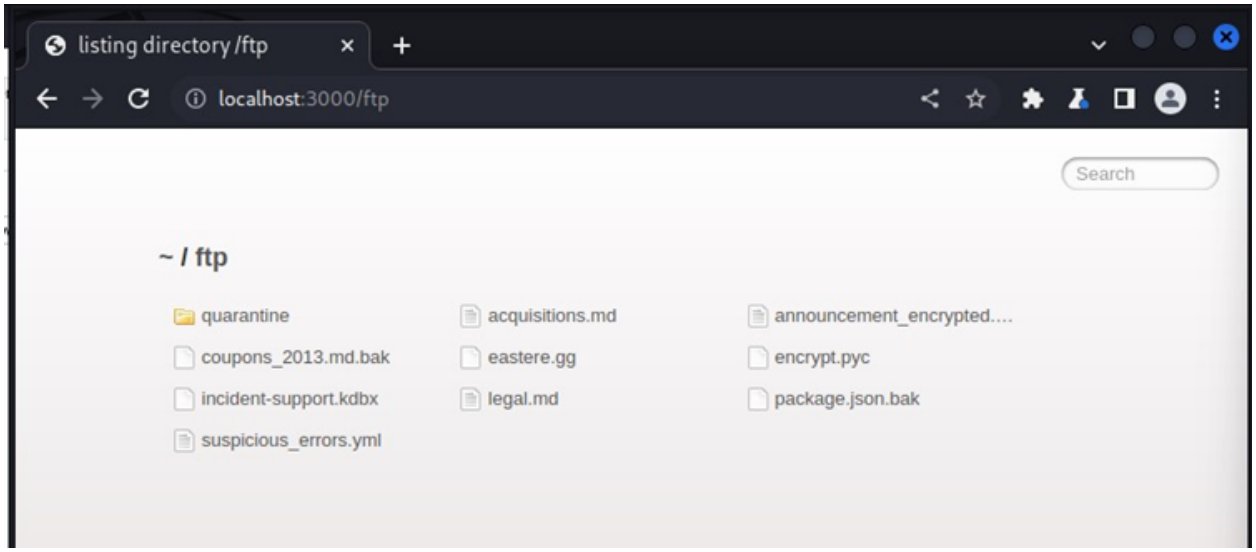
Every website has a reference page called *robots.txt* that declares the names of files/folders that should not be accessed by the browser.

We should try visiting robots.txt of juice-shop

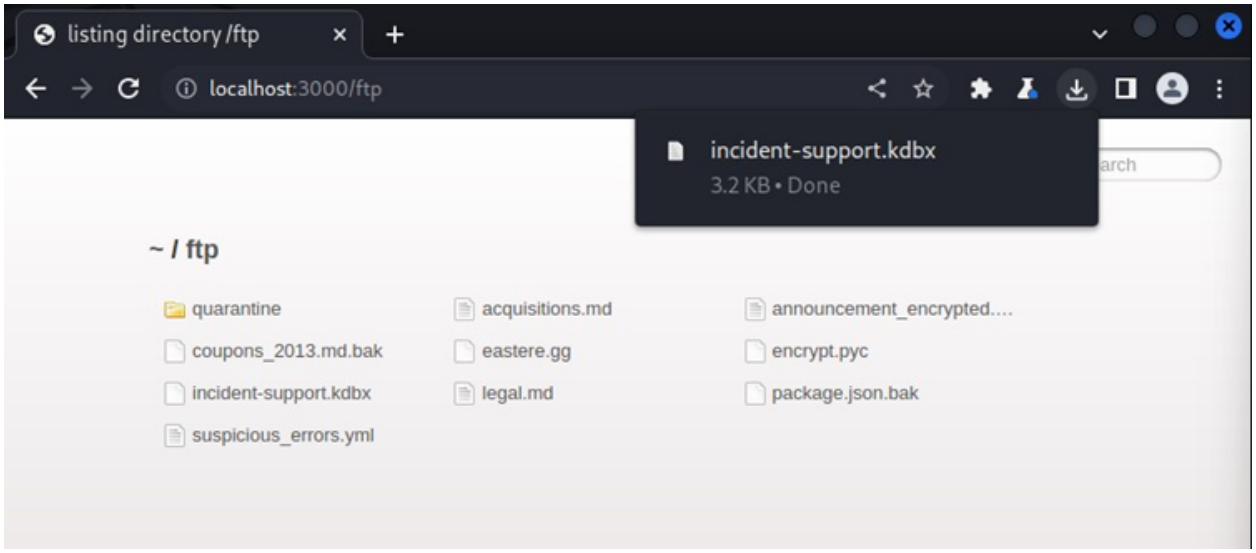


We can see no users are allowed to access the /ftp folder. So lets try accessing it from the browser url bar.

If We Try to access localhost:3000/ftp , we get a list of files and folders.

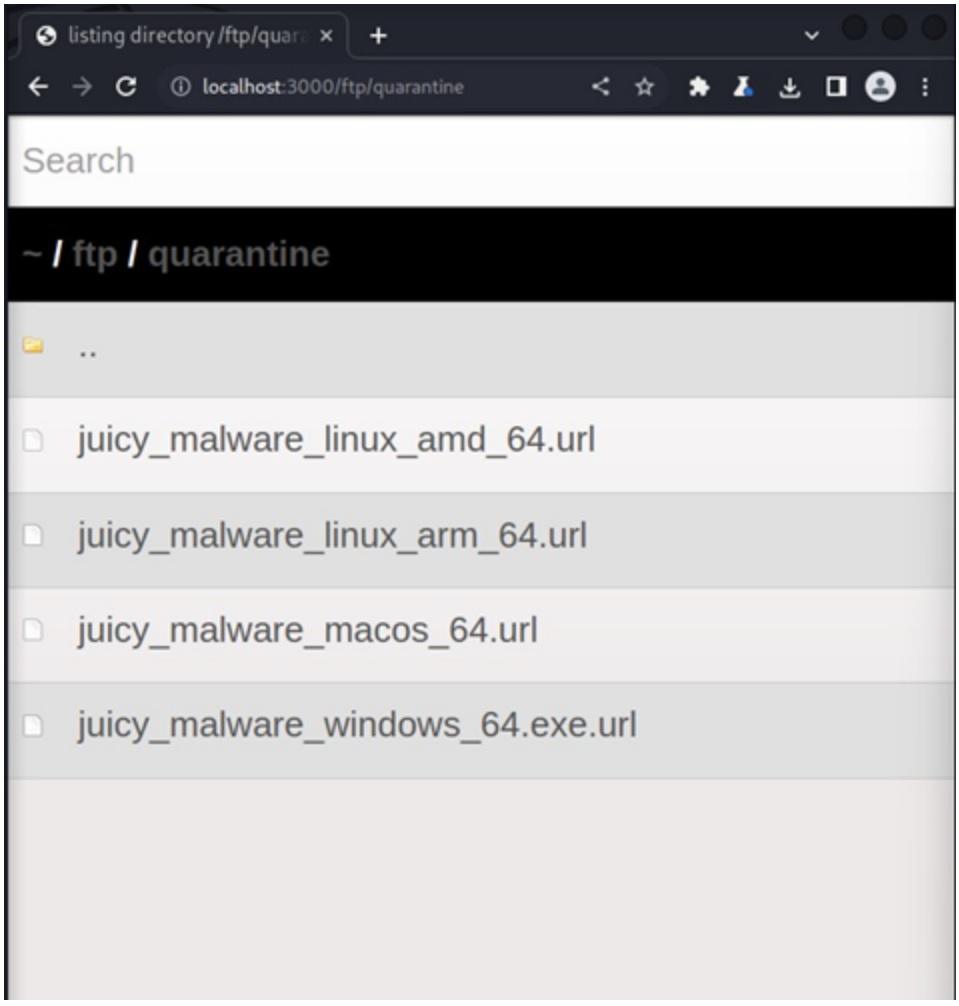


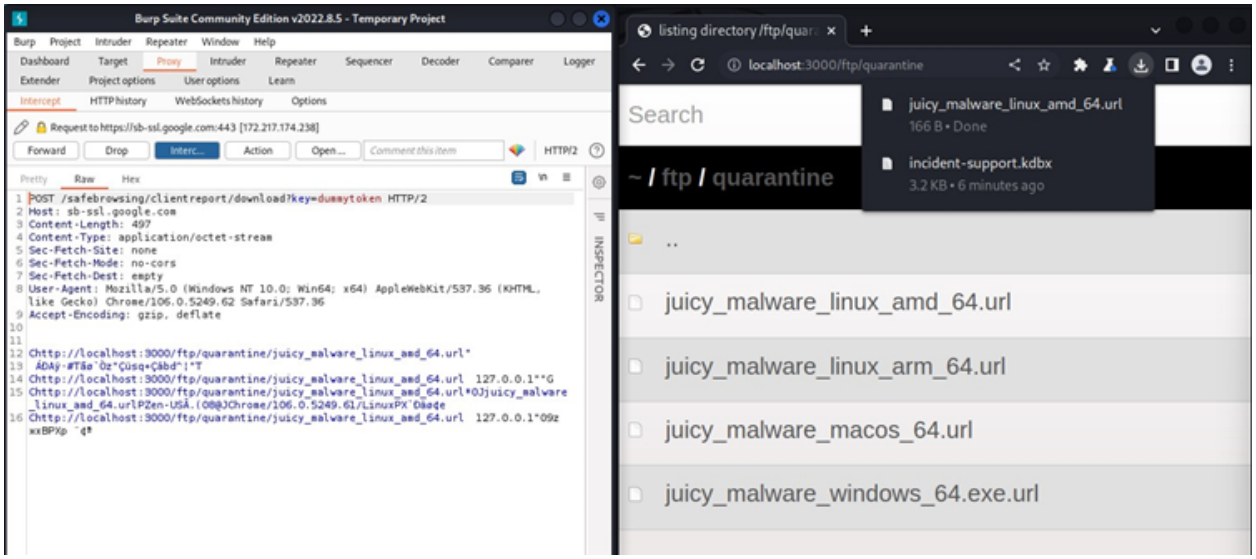
Lets try accessing a random file.



As we can see we can download incident-support.kdbx

If we try to access qurantine folder, using interceptor.


listing directory /ftp/quar
localhost:3000/ftp/quarantine
Search
~ / ftp / quarantine
..
juicy_malware_linux_amd_64.url
juicy_malware_linux_arm_64.url
juicy_malware_macos_64.url
juicy_malware_windows_64.exe.url

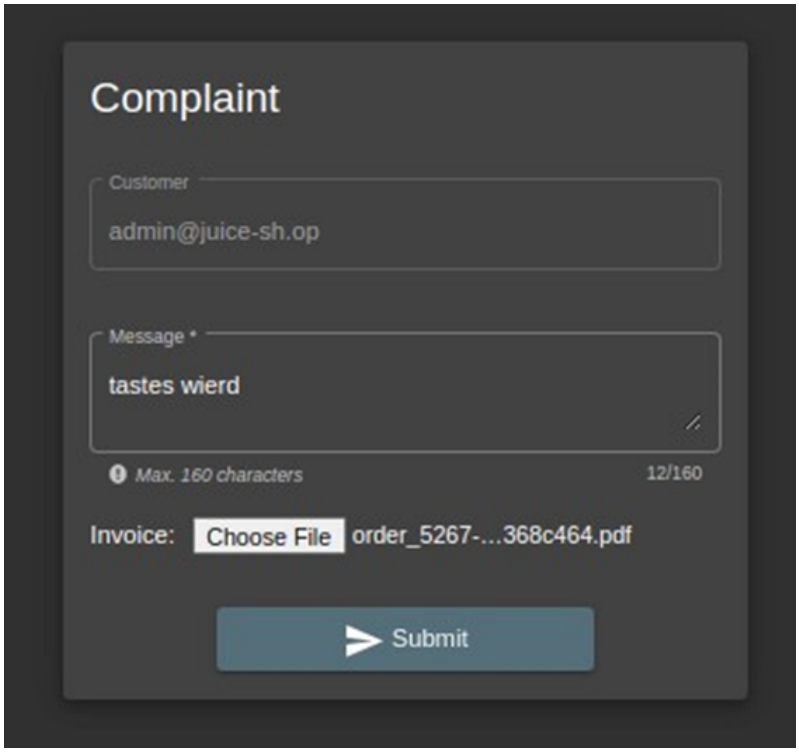

Burp Suite Community Edition v2022.8.5 - Temporary Project
POST /safebrowsing/clientreport/download?key=dummytoken HTTP/2
Host: sb-ssl.google.com
Content-Length: 497
Content-Type: application/octet-stream
Sec-Fetch-Site: none
Sec-Fetch-Mode: no-cors
Sec-Fetch-Dest: empty
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36
Accept-Encoding: gzip, deflate
http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url
http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url 127.0.0.1
http://localhost:3000/ftp/quarantine/juicy_malware_linux_amd_64.url 127.0.0.1
listing directory /ftp/quar
localhost:3000/ftp/quarantine
Search
juicy_malware_linux_amd_64.url
166 B • Done
incident-support.kdbx
3.2 KB • 6 minutes ago
~ / ftp / quarantine
..
juicy_malware_linux_amd_64.url
juicy_malware_linux_arm_64.url
juicy_malware_macos_64.url
juicy_malware_windows_64.exe.url

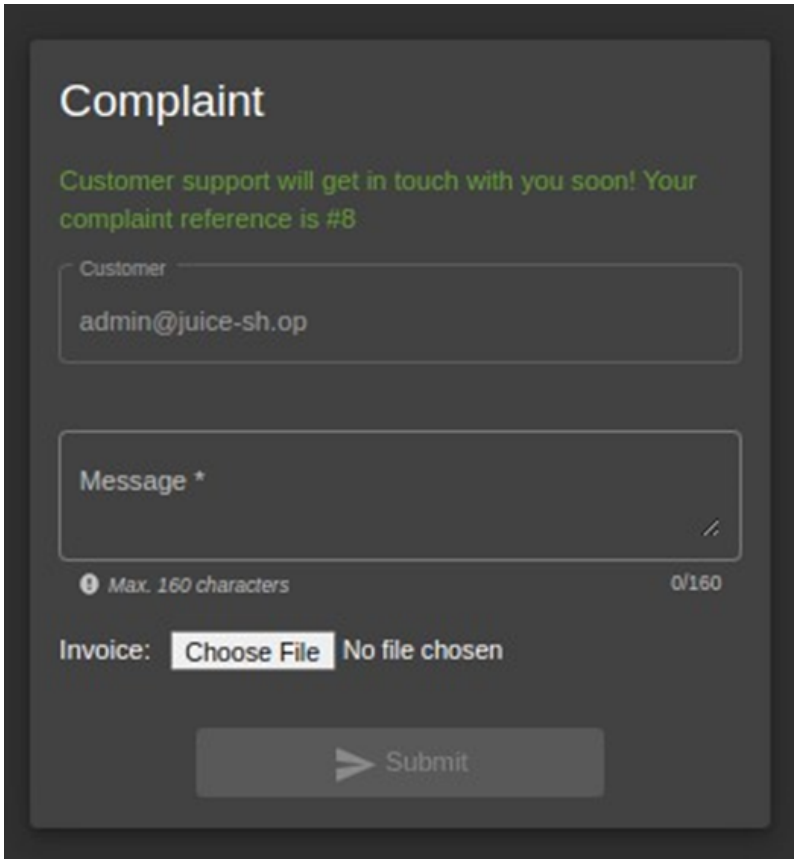
9) Insecure Design

CWE-657: Violation of Secure Design Principles
Description: The product violates well-established principles for secure design.
Business Impact: Insecure design within a business's systems and processes can have far-reaching and detrimental impacts. This includes vulnerabilities in software, network architecture, or physical security measures. First and foremost, such flaws can lead to data breaches, exposing sensitive customer information and intellectual property. The financial consequences are substantial, encompassing the costs of breach remediation, regulatory fines, and potential lawsuits. Moreover, the damage to the company's reputation can be severe, eroding customer trust and loyalty. This erosion of trust can result in decreased sales, a loss of existing clients, and difficulty in attracting new ones. Insecure design can also disrupt operations, causing downtime, which translates to lost productivity and revenue. Over time, the cumulative impact of these issues can impair the overall competitiveness and viability of the business in the marketplace. To mitigate these risks, businesses must prioritize robust security measures and adopt a proactive approach to identifying and rectifying vulnerabilities in their design and infrastructure.

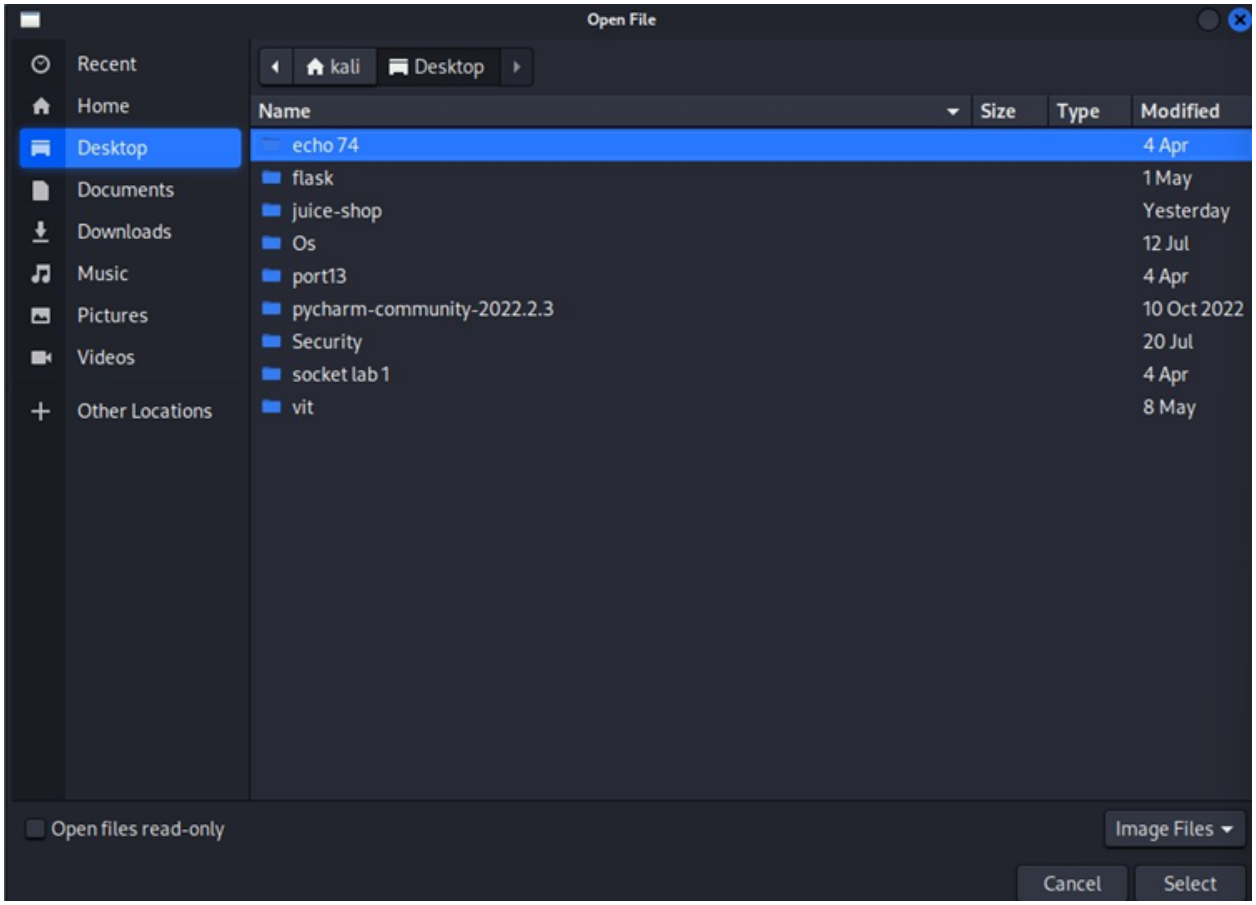
Testing:

Let's consider a situation where we can upload a file with malicious code into the server. We notice that there is a complaint upload portal where we can upload a file. What if we figure out a way to install a backdoor through that upload. Giving the upload section a closer look, we can figure out that the file upload needs to be a pdf/zip file . We can easily figure out that the upload will accept any file with extension .pdf or .zip . Lets first try uploading a .pdf file


Complaint
Customer
admin@juice-sh.op
Message *
tastes wierd
Max. 160 characters
12/160
Invoice:
Choose File
order_5267-...368c464.pdf
Submit

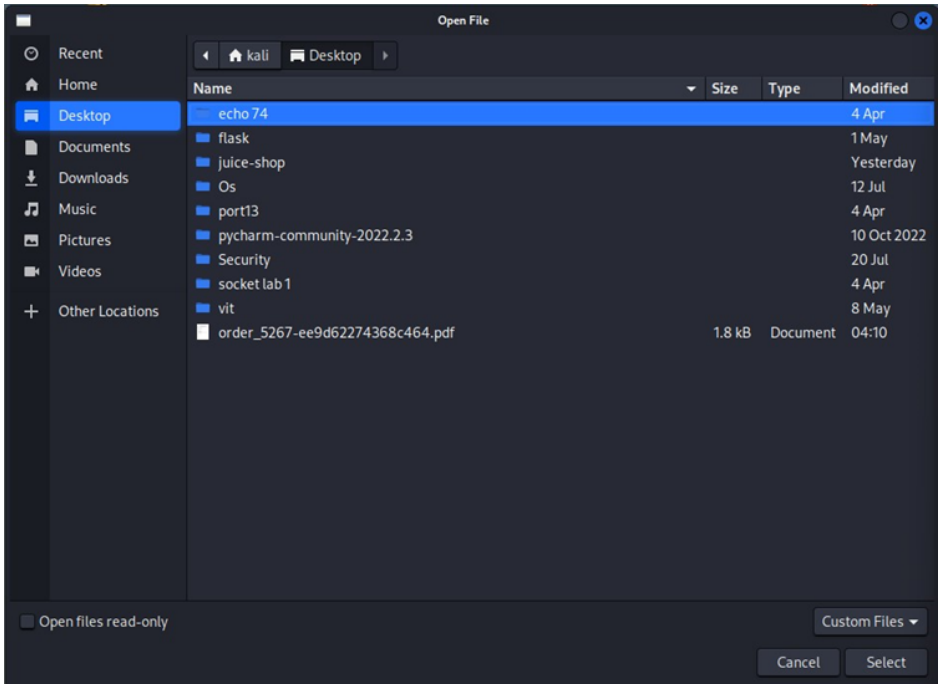

Complaint
Customer support will get in touch with you soon! Your complaint reference is #8
Customer
admin@juice-sh.op
Message *
Max. 160 characters
0/160
Invoice:
Choose File
No file chosen
Submit

The pdf document gets uploaded. Lets try uploading a file with some other extension.



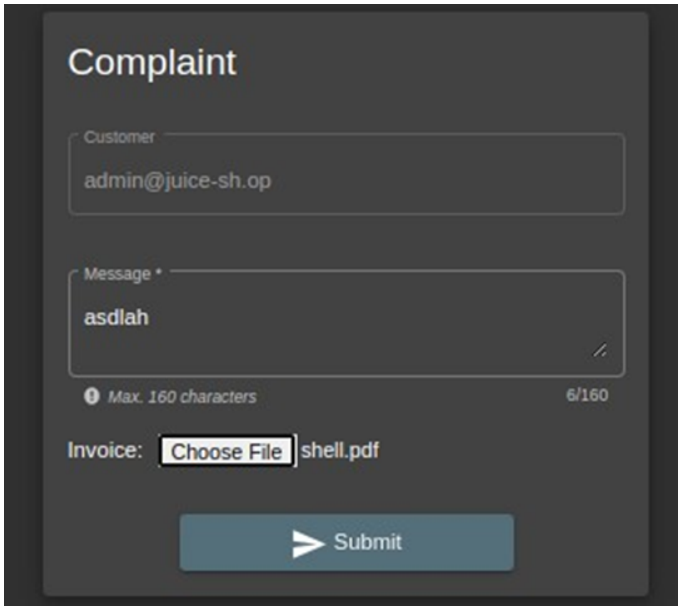
As we can see, files with extensions other than .pdf / .zip are not accessible but we do have other files with different extensions in this particular folder.

In the desktop folder I have two files with .txt extension, but the choose file section is overseeing those files. So we need to find an alternate solution. We can save the malicious file with the .pdf extension and try uploading it.

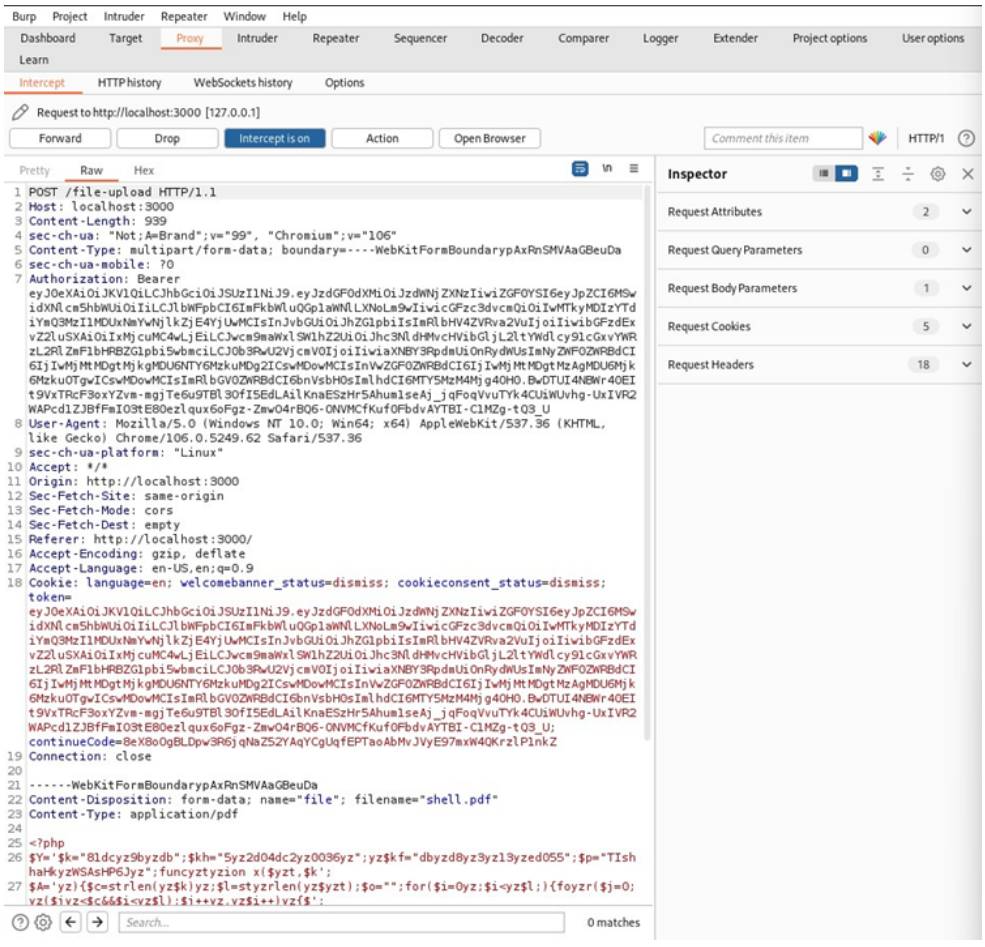


What if we build a malicious file with extension .pdf? Although the content in the pdf file will be malicious, it won't be affecting the server as without the extension it is important to denote the language in which the code should be executed.

Lets try uploading the malicious code as a .pdf extension, intercept it and then change the value to .php later. I used weevely to create a php backdoor and later changed the extension to .pdf later.



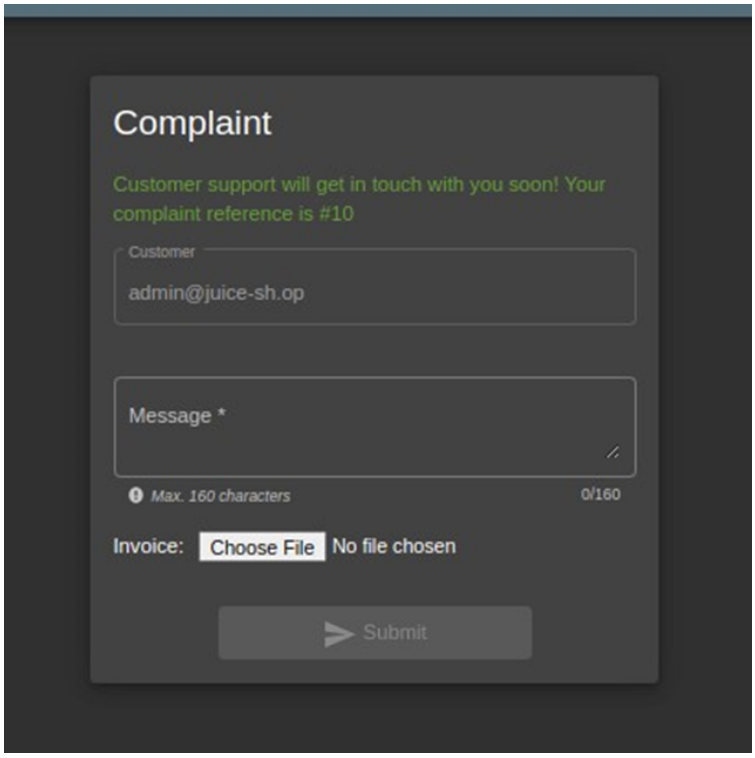
Now I turn the intercept on and press the submit button.



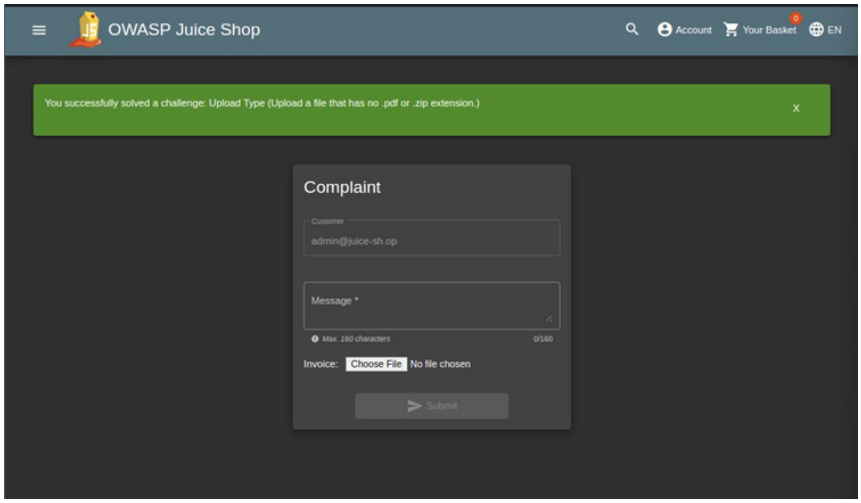
This is the request and we can notice the filename is shell.pdf and file type is application/pdf but to make it executable, we need to change the extension to .php . So now I change the extension to .php .

```
vZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy
zL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZW
6IjIwMjMtMDgtMjkgMDU6NTY6MzkuMDg2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMz
6MzkuOTgwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzM4Mjg4OH0.BwDTUI
t9VxTRcF3oxYZvm-mgjTe6u9TBl30fI5EdLAilKnaESzHr5Ahum1seAj_jqFoqVvuTYk4CUiWUv
WAPcdlZJBfFmIO3tE80ezlqux6oFgz-Zmw04rBQ6-ONVMCfKuf0FbdvAYTBI-ClMZg-tQ3_U;
continueCode=8eX8oOgBLDpw3R6jqNaZ52YAqYCgUqfEPTaoAbMvJVyE97mxW4QKrzlP1nkZ
Connection: close

------WebKitFormBoundarypAxRnSMVAaGBeuDa
Content-Disposition: form-data; name="file"; filename="shell.php"
Content-Type: application/pdf

<?php
```

Now we forward the request.



As we can see the request has been submitted with extension .php . And hence we've successfully injected a backdoor.



10) BROKEN ACCESS CONTROL (Different method)

CWE-285: Improper Authorization

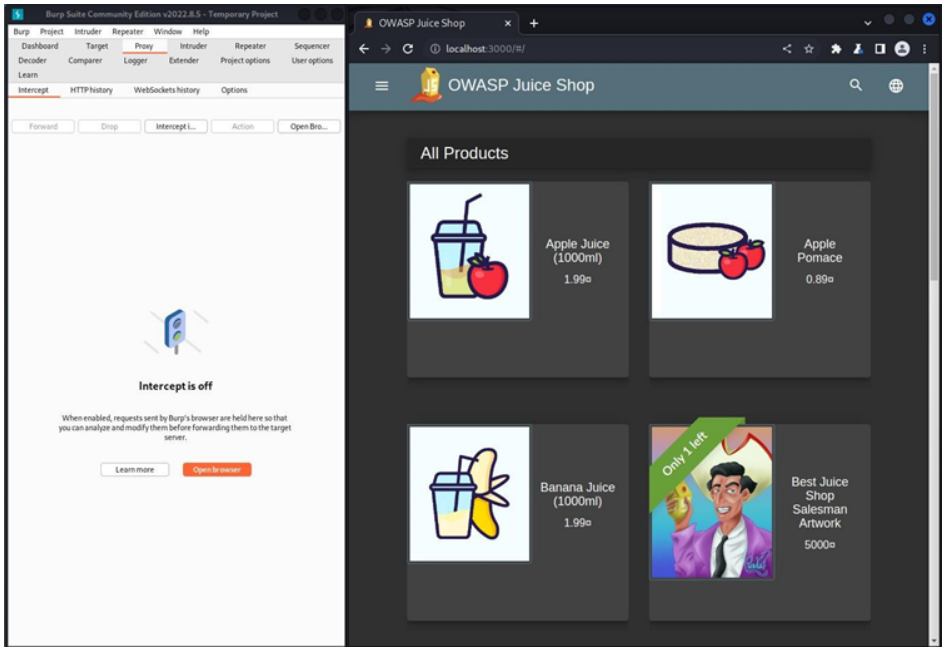
Description: The product does not perform or incorrectly performs an authorization check when an actor attempts to access a resource or perform an action.

Business Impact: Broken access control within a business's systems and applications can have significant and detrimental consequences. When users can gain unauthorized access to sensitive data or functions, it can result in data breaches, potentially exposing confidential information, customer records, or intellectual property. This may lead to financial losses, including the costs of breach investigation, regulatory fines, and legal liabilities. Additionally, it can erode customer trust and damage the company's reputation, potentially resulting in lost customers and reduced revenue. Furthermore, such security lapses can disrupt business operations, leading to downtime, lost productivity, and, in some cases, even the complete compromise of critical systems. Businesses should address

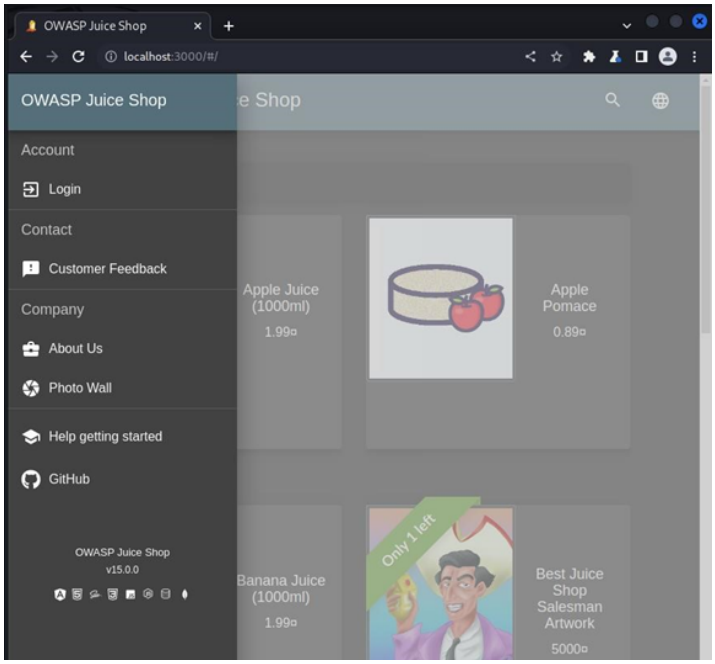
broken access control promptly and implement proper access management to mitigate these risks and protect their data, reputation, and bottom line.

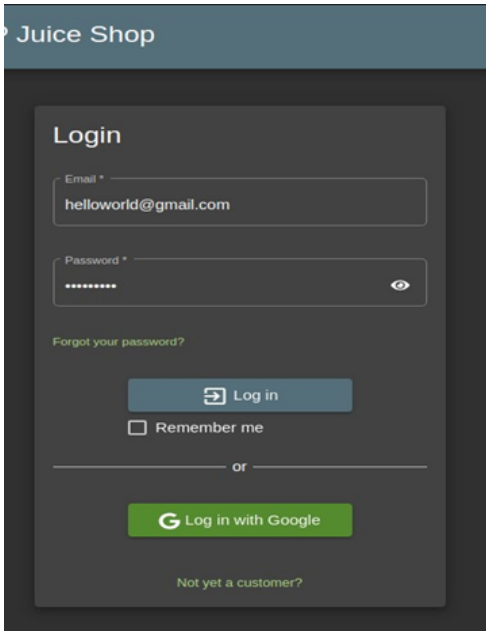
Testing:

Launch the BurpSuit App and go to the proxy tab to launch the burpsuit browser. In the browser access the juice shop website with the localhost:3000 address.



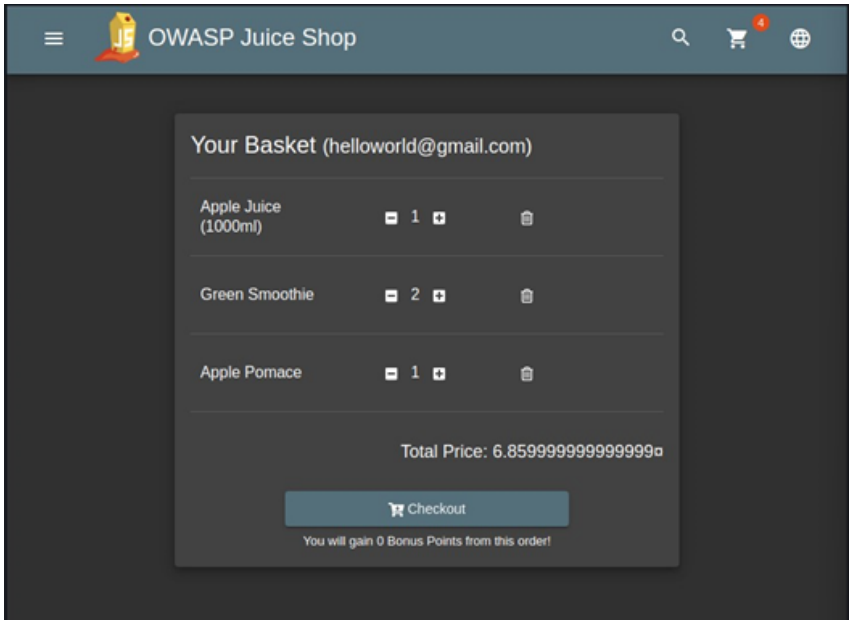
Login to the website with relevant credentials



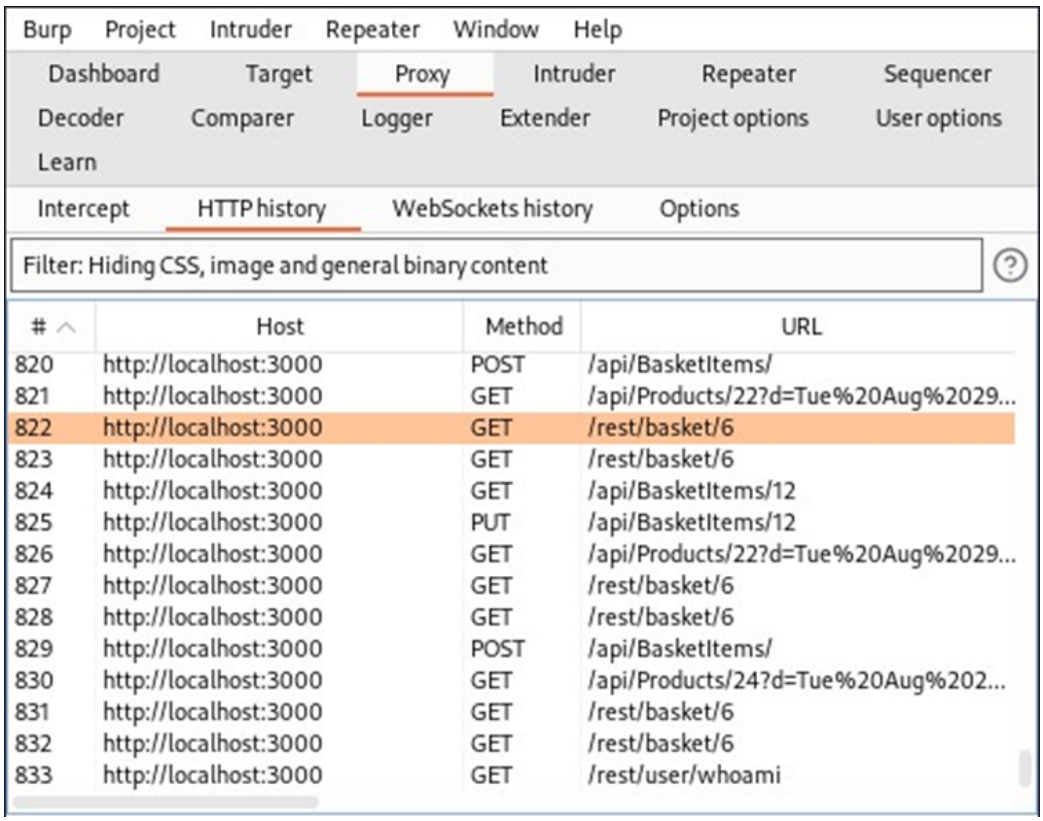


After completing the login process, add drinks of your choice to the cart.

For this particular instance, we will go for -> 1 Apple Juice 2 Green Smoothie 1 Apple Pomace



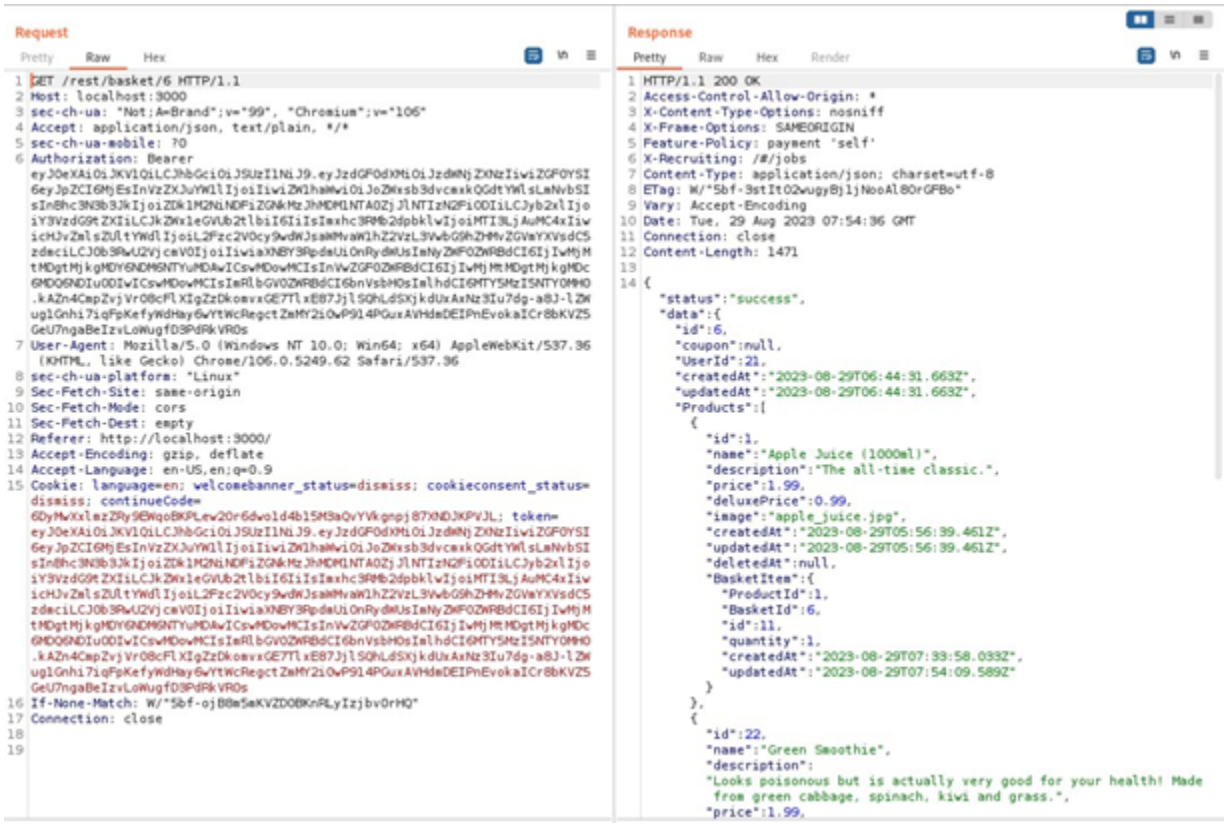
Now go to the http history section in proxy tab and try to find the request that was meant for creation of your basket It would be a get request with basket/<Cart number> URL



We can see that this is the URL linked to our basket.

If we check the Request panel, which must be set in the raw mode, we would find some data related to our basket and items in it

If we select that data and redirect it to the repeater, and click on send, we get some response in JSON format. If we have a closer Look....



These are the data related to the items that we ordered,

Apple Juice


```
"id":1,
"name":"Apple Juice (1000ml)",
"description":"The all-time classic.",
"price":1.99,
"deluxePrice":0.99,
"image":"apple_juice.jpg",
"createdAt":"2023-08-29T05:56:39.461Z",
"updatedAt":"2023-08-29T05:56:39.461Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":1,
  "BasketId":6,
  "id":11,
  "quantity":1,
  "createdAt":"2023-08-29T07:33:58.033Z",
  "updatedAt":"2023-08-29T07:54:09.589Z"
```

Green Smoothie


```
"id":22,
"name":"Green Smoothie",
"description":
"Looks poisonous but is actually very good for your health! Made
 from green cabbage, spinach, kiwi and grass.",
"price":1.99,
"deluxePrice":1.99,
"image":"green_smoothie.jpg",
"createdAt":"2023-08-29T05:56:39.462Z",
"updatedAt":"2023-08-29T05:56:39.462Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":22,
  "BasketId":6,
  "id":12,
  "quantity":2,
  "createdAt":"2023-08-29T07:34:12.176Z",
  "updatedAt":"2023-08-29T07:34:14.177Z"
```

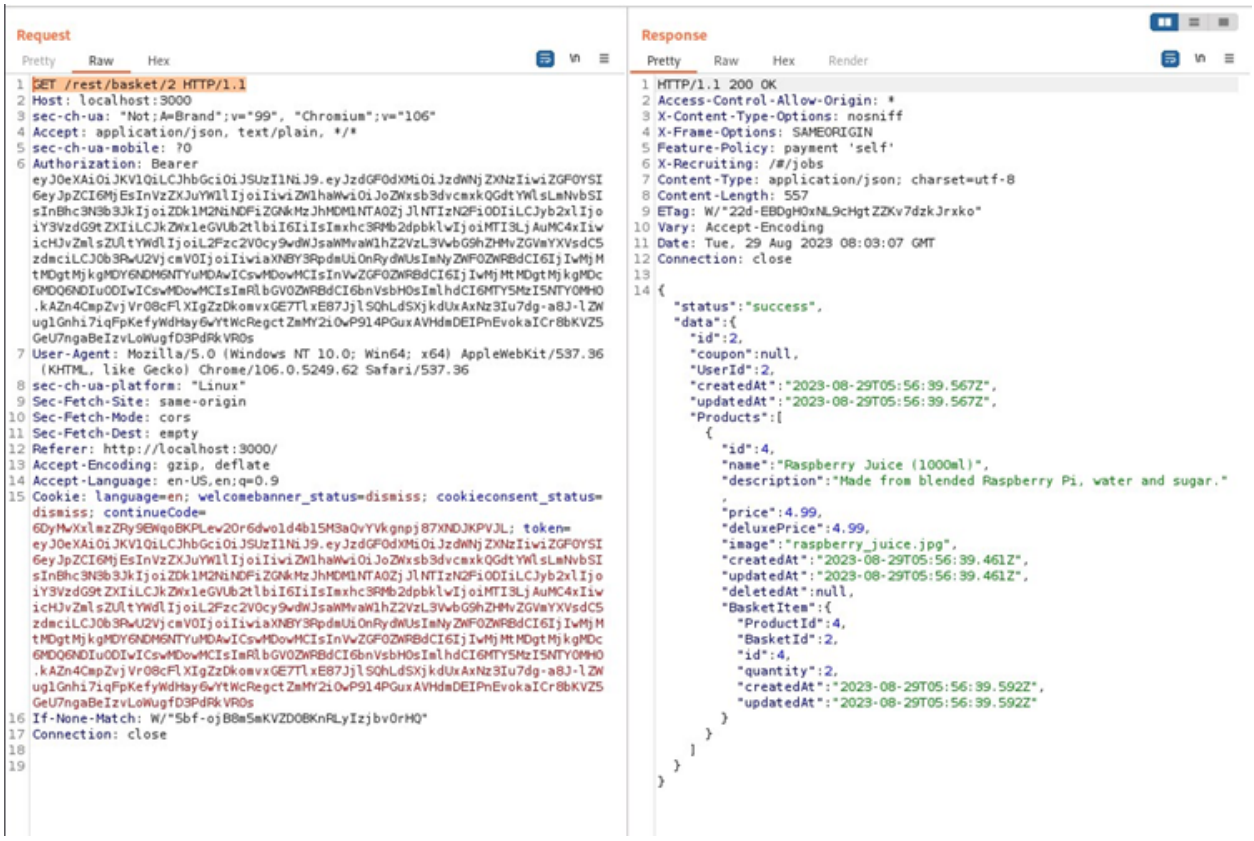

Apple Pomace


```
"id":24,
"name":"Apple Pomace",
"description":
"Finest pressings of apples. Allergy disclaimer: Might contain t
races of worms. Can be <a href=\"/#recycle\">sent back to us</a>
 for recycling.",
"price":0.89,
"deluxePrice":0.89,
"image":"apple_pressings.jpg",
"createdAt":"2023-08-29T05:56:39.462Z",
"updatedAt":"2023-08-29T05:56:39.462Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":24,
  "BasketId":6,
  "id":13,
  "quantity":1,
  "createdAt":"2023-08-29T07:34:22.172Z",
  "updatedAt":"2023-08-29T07:34:22.172Z"
```


GET /rest/basket/6 HTTP/1.1 is somehow linked to our basket and if it is so, most probably the digit 6 is our basket ID.

Now what if I try to change the basket Id to 2 in the request section? This may or may not change the response but it is worth giving a try, if there is a change, we will be able to declare a Broken access control vulnerability.

Now when I changed the request to GET /rest/basket/2 HTTP/1.1 the response has significantly changed.



Request

Pretty Raw Hex

```
1 GET /rest/basket/2 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 Authorization: Bearer
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI
  6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSI
  sInBhc3N3b3JkIjoiZDk1M2NiNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2FiODIiLCJyb2xlIjo
  iY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiw
  icHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5
  zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjM
  tMDgtMjkgMDY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
  6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0
  .kAZn4CmpZvjVr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZW
  uglGnhi7iqFpKefyWdHay6wYtWcRegctZmMY2iOwP914PGuxAVHdmDEIPnEvokaICr8bKVZ5
  GeU7ngaBeIzvLoWugfD3PdRkVROs
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  (KHTML, like Gecko) Chrome/106.0.5249.62 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=
  dismiss; continueCode=
  6DyMwXxlmzZRy9EWqoBKPLew2Or6dwold4b15M3aQvYVkgnpj87XNDJKPVJL; token=
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI
  6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSI
  sInBhc3N3b3JkIjoiZDk1M2NiNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2FiODIiLCJyb2xlIjo
  iY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiw
  icHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2VzL3VwbG9hZHMvZGVmYXVsdC5
  zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjM
  tMDgtMjkgMDY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
  6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0
  .kAZn4CmpZvjVr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZW
  uglGnhi7iqFpKefyWdHay6wYtWcRegctZmMY2iOwP914PGuxAVHdmDEIPnEvokaICr8bKVZ5
  GeU7ngaBeIzvLoWugfD3PdRkVROs
16 If-None-Match: W/"5bf-ojB8m5mKVZDOBKnRLyIzjbvOrHQ"
17 Connection: close
18
19
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: application/json; charset=utf-8
8 Content-Length: 557
9 ETag: W/"22d-EBDgHOxNL9cHgtZZXv7dzkJrxko"
10 Vary: Accept-Encoding
11 Date: Tue, 29 Aug 2023 08:03:07 GMT
12 Connection: close
13
14 {
    "status":"success",
    "data":{
      "id":2,
      "coupon":null,
      "UserId":2,
      "createdAt":"2023-08-29T05:56:39.567Z",
      "updatedAt":"2023-08-29T05:56:39.567Z",
      "Products":[
        {
          "id":4,
          "name":"Raspberry Juice (1000ml)",
          "description":"Made from blended Raspberry Pi, water and sugar."
          ,
          "price":4.99,
          "deluxePrice":4.99,
          "image":"raspberry_juice.jpg",
          "createdAt":"2023-08-29T05:56:39.461Z",
          "updatedAt":"2023-08-29T05:56:39.461Z",
          "deletedAt":null,
          "BasketItem":{
            "ProductId":4,
            "BasketId":2,
            "id":4,
            "quantity":2,
            "createdAt":"2023-08-29T05:56:39.592Z",
            "updatedAt":"2023-08-29T05:56:39.592Z"
          }
        }
      ]
    }
}
```

We can notice that the request tab holds basket id 2 and the response tab is giving response 200 OK which means we have successfully accessed the basket with ID 2

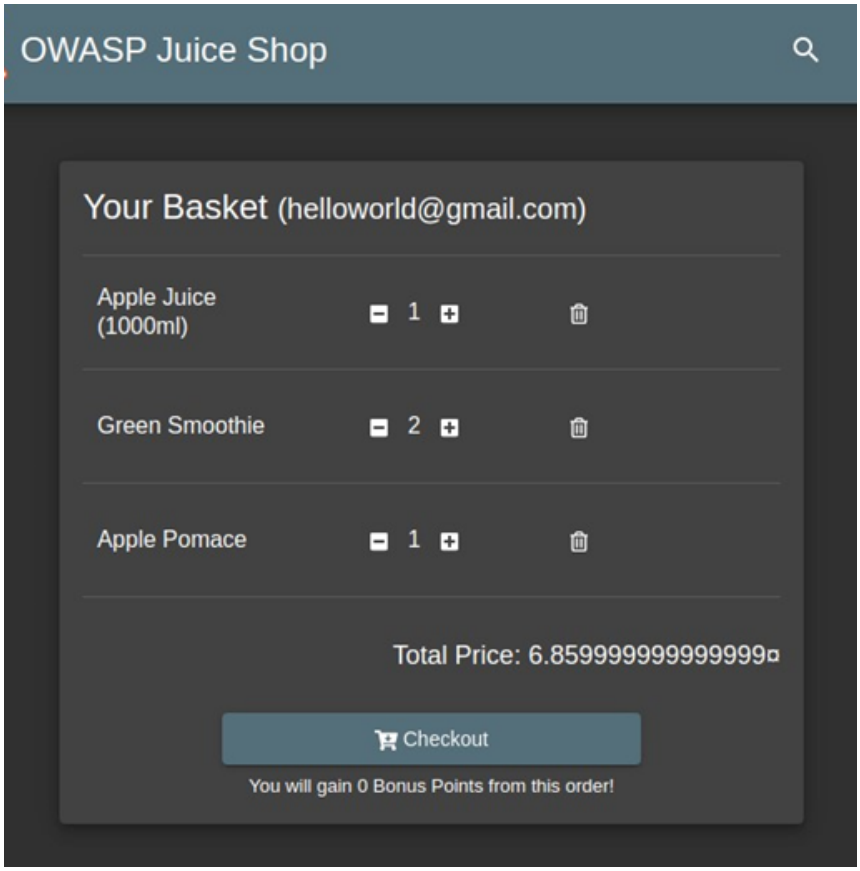
Let's see if that's actually different from the basket that we built.

```
"id":4,
"name":"Raspberry Juice (1000ml)",
"description":"Made from blended Raspberry Pi, water and sugar."
,
"price":4.99,
"deluxePrice":4.99,
"image":"raspberry_juice.jpg",
"createdAt":"2023-08-29T05:56:39.461Z",
"updatedAt":"2023-08-29T05:56:39.461Z",
"deletedAt":null,
"BasketItem":{
  "ProductId":4,
  "BasketId":2,
  "id":4,
  "quantity":2,
  "createdAt":"2023-08-29T05:56:39.592Z",
  "updatedAt":"2023-08-29T05:56:39.592Z"
```

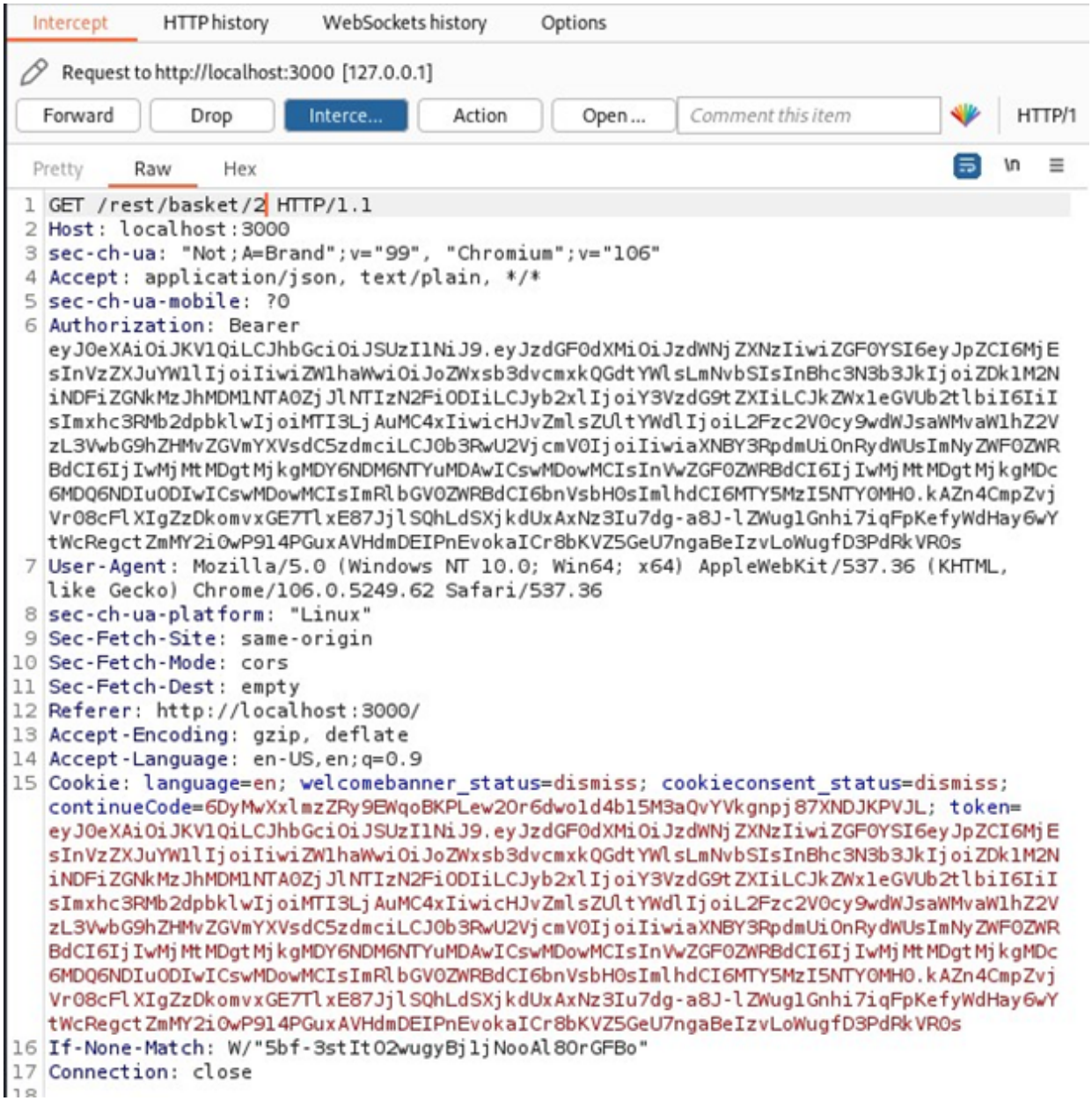
We can see that there is an item named Raspberry Juice which we didn't order. This proves that we have accessed some other basket.

Now we should try to implement it using the intercept function.

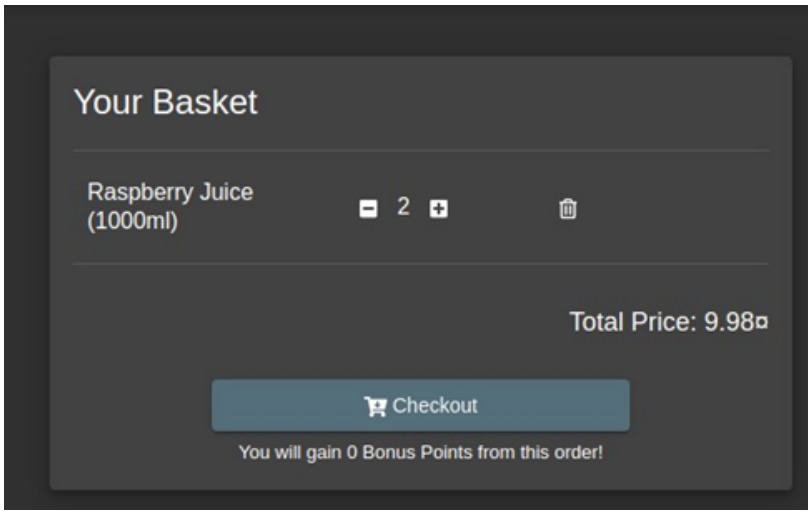
If we go back to the website, we see our very own cart that we built, but now if we turn on the intercept and traverse 1 directory back and then retry to get into our basket by changing the request id from 6 to 2, we must notice a change in our basket.



Basket with order ID 6.


Intercept
HTTP history
WebSockets history
Options
Request to http://localhost:3000 [127.0.0.1]
Forward
Drop
Interce...
Action
Open ...
Comment this item
HTTP/1
Pretty
Raw
Hex
\n
1 GET /rest/basket/2 HTTP/1.1
2 Host: localhost:3000
3 sec-ch-ua: "Not;A=Brand";v="99", "Chromium";v="106"
4 Accept: application/json, text/plain, */*
5 sec-ch-ua-mobile: ?0
6 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjE
sInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiZDk1M2N
iNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2Fi0DIiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiI
sImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2V
zL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWR
BdCI6IjIwMjMtMDgtMjkgMjY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0.kAZn4CmpZvj
Vr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZWuglGnhi7iqFpKefyWdHay6wY
tWcRegctZmMY2i0wP914PGuxAVHdmDEIPnEvokaICr8bKVZ5GeU7ngaBeIzvLoWugfD3PdRkVR0s
7 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/106.0.5249.62 Safari/537.36
8 sec-ch-ua-platform: "Linux"
9 Sec-Fetch-Site: same-origin
10 Sec-Fetch-Mode: cors
11 Sec-Fetch-Dest: empty
12 Referer: http://localhost:3000/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss;
continueCode=6DyMwXxlmzZRy9EWqoBKPLew2Or6dwold4bl5M3aQvYVkgnpj87XNDJKPVJL; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjE
sInVzZXJuYW1lIjoiIiwiZW1haWwiOiJoZWxsb3dvcmxkQGdtYWlsLmNvbSIsInBhc3N3b3JkIjoiZDk1M2N
iNDFiZGNkMzJhMDM1NTA0ZjJlNTIzN2Fi0DIiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiI
sImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiL2Fzc2V0cy9wdWJsaWMvaW1hZ2V
zL3VwbG9hZHMvZGVmYXVsdC5zdmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWR
BdCI6IjIwMjMtMDgtMjkgMjY6NDM6NTYuMDAwICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjMtMDgtMjkgMDc
6MDQ6NDIuODIwICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTY5MzI5NTY0MH0.kAZn4CmpZvj
Vr08cFlXIgZzDkomvxGE7TlxE87JjlSQhLdSXjkdUxAxNz3Iu7dg-a8J-lZWuglGnhi7iqFpKefyWdHay6wY
tWcRegctZmMY2i0wP914PGuxAVHdmDEIPnEvokaICr8bKVZ5GeU7ngaBeIzvLoWugfD3PdRkVR0s
16 If-None-Match: W/"5bf-3stIt02wugyBj1jNooAl8OrGFBo"
17 Connection: close

Now we forward the request to the browser and see what result we get



We can see Our basket has changed and the order includes Raspberry juice

Hence we can declare a Broken access control Vulnerability as I am able to access Baskets of other clients.