



Raport z oceny bezpieczeństwa aplikacji OWASP Juice Shop

Raport przygotowany dla Bezpierczny Kod ("BK")

Warszawa, 24.09.2023

Historia wersji

Wersja	Data	Opis zmiany	Autor
1	24/09/2023	Publikacja oficjalnej pierwszej wersji raportu.	Gabriela Czarnecka

Punkty kontaktowe

Firma	Imię i nazwisko	E-mail
Bezpieczny Kod	Osoba kontaktowa	andrzej@bezpiecznykod.pl
Wykonawca	Gabriela Czarnecka	czarneckagabriela8@gmail.com

Spis treści

Historia wersji	2
Punkty kontaktowe	2
1. Podsumowanie kierownicze	5
2. Cel i zakres prac	7
3. Metodyka i użyte standardy	8
4. Podsumowanie rezultatów prac	9
5. Rezultaty prac	11
(P1- CRITICAL) 5.1 Możliwość zalogowania się na konto administratora	12
Podsumowanie	12
Warunki niezbędne do wykorzystania podatności	12
Szczegóły techniczne	12
Lokalizacja problemu	14
Rekomendacje	14
(P2- HIGH) 5.2 Możliwość przejęcia konta użytkownika poprzez uzyskanie ciasteczek sesyjnych	15
Podsumowanie	15
Warunki niezbędne do wykorzystania podatności	15
Szczegóły techniczne	15
Lokalizacja problemu	16
Rekomendacje	16
(P3- MEDIUM) 5.3 Możliwość podglądania koszyków innych użytkowników	17
Podsumowanie	17
Warunki niezbędne do wykorzystania podatności	17
Szczegóły techniczne	17
Lokalizacja problemu	19
Rekomendacje	19
(P3- MEDIUM) 5.4 Możliwość modyfikowania zawartości koszyków innych użytkowników	20
Podsumowanie	20
Warunki niezbędne do wykorzystania podatności	20
Szczegóły techniczne	20
Lokalizacja problemu	22
Rekomendacje	22
(P3- MEDIUM) 5.5 Możliwość uzyskania dostępu do zasobów serwera aplikacji	23
Podsumowanie	23
Warunki niezbędne do wykorzystania podatności	23
Szczegóły techniczne	23
Lokalizacja problemu	25
Rekomendacje	25



(P3- MEDIUM) 5.6 Nieautoryzowany dostęp do katalogu ftp	26
Podsumowanie	26
Warunki niezbędne do wykorzystania podatności	26
Szczegóły techniczne	26
Lokalizacja problemu	26
Rekomendacje	27
(P5- INFORMATIONAL) 5.7 Słaba implementacja mechanizmu przypominania hasła	28
Podsumowanie	28
Warunki niezbędne do wykorzystania podatności	28
Szczegóły techniczne	28
Lokalizacja problemu	29
Rekomendacje	29
(P5- INFORMATIONAL) 5.8 Słaba polityka haseł	30
Podsumowanie	30
Warunki niezbędne do wykorzystania podatności	30
Szczegóły techniczne	30
Lokalizacja problemu	31
Rekomendacje	31
(P5- INFORMATIONAL) 5.9 Komunikat umożliwiający enumerację użytkowników	32
Podsumowanie	32
Warunki niezbędne do wykorzystania podatności	32
Szczegóły techniczne	32
Lokalizacja problemu	33
Rekomendacje	33
(P5- INFORMATIONAL) 5.10 Wykorzystanie biblioteki z krytycznymi podatnościami	34
Podsumowanie	34
Warunki niezbędne do wykorzystania podatności	34
Szczegóły techniczne	34
Lokalizacja problemu	35
Rekomendacje	35
INNE NIEPRAWIDŁOWOŚCI	36



1. Podsumowanie kierownicze

Dokument jest podsumowaniem prac wykonanych przez Gabrielę Czarnecką dla **Bezpieczny Kod** na podstawie oferty z dnia 23.08.2023r.

W skład oceny bezpieczeństwa wchodziły:

- Testy bezpieczeństwa.

Obiektem oceny była aplikacja OWASP Juice Shop w wersji 15.0.0 dostępna pod adresami:

- <http://localhost:3000>

Główne problemy wykryte podczas oceny bezpieczeństwa dotyczą OWASP Juice Shop:

- brak sanityzacji oraz walidacji danych przekazywanych przez użytkowników, a także mechanizmów opartych na tzw. "Allow-list", co umożliwia na różnego rodzaju wstrzyknięcia,
- brak odpowiednich mechanizmów autoryzacji użytkowników,
- słaba polityka haseł, słaby mechanizm przypomnienia hasła,
- podatności w używanych komponentach,
- komunikaty umożliwiające enumeracje użytkowników.

Łącznie zostało znalezionych 10 problemów, które sklasyfikowane zostały na podstawie VRT (Bugcrowd's Vulnerability Rating Taxonomy), w tym:

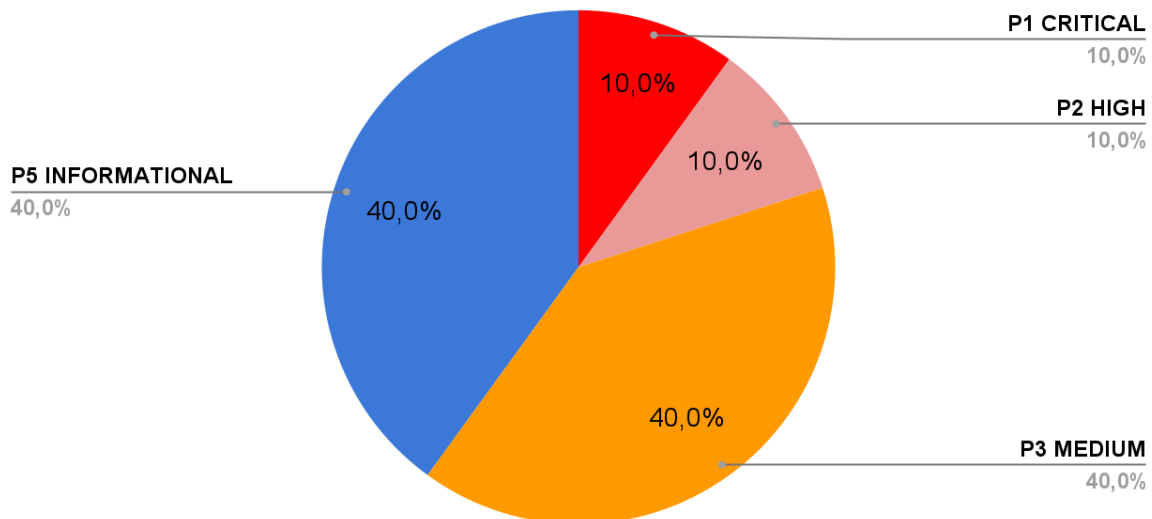
- 1 problem o największej krytyczności,
- 1 problem o krytyczności wysokiej,
- 4 problemy o krytyczności średniej,
- 4 problemy o krytyczności informacyjnej,

oraz ujawniono jedną nieprawidłowość o krytyczności informacyjnej, która została opisana w sekcji "Inne nieprawidłowości" rozdziału 5.



Rozbicie procentowe ze względu na krytyczność znalezionych problemów wygląda następująco:

Procentowy udział znalezionych nieprawidłowości wg ich krytyczności



Ocena bezpieczeństwa została wykonana podejściem białej skrzynki (white-box) według standardu OWASP Application Security Verification Standard v4 poziomu 1 obejmującego również podatności z listy OWASP Top 10 2021.



2. Cel i zakres prac

Celem prac była ocena bezpieczeństwa aplikacji OWASP Juice Shop po to, aby zidentyfikować ryzyka i zagrożenia, które mogą wpłynąć na podstawowe właściwości bezpieczeństwa systemów IT: Poufność (confidentiality), integralność (integrity) oraz dostępność (availability).

Zakres prac obejmował:

- Przeprowadzenie manualnych testów bezpieczeństwa aplikacji pod kątem standardu OWASP Application Security Verification Standard v4 poziomu 1.

Ocena została wykonana podejściem białej skrzynki (white-box).

Do testów zostały oddane:

- Aplikacja OWASP Juice Shop w wersji 15.0.0 dostępna pod adresami:
 - `http://localhost:3000`

Podczas prac nie napotkaliśmy żadnych ograniczeń ze strony wytwórcy oprogramowania.



3. Metodyka i użyte standardy

Metodyka zastosowana w trakcie testów bezpieczeństwa aplikacji opiera się o najlepsze rynkowe praktyki, w tym o metodykę OWASP Web Security Testing Guide v4.

Ocena bezpieczeństwa systemu została wykonana pod kątem OWASP Application Security Verification Standard v4 poziomu 1, który obejmuje również podatności z listy OWASP Top 10 2021.

W pracach oparliśmy się również o standardy organizacji takich jak NIST, ENISA czy CIS.

4. Podsumowanie rezultatów prac

Prace zawarte były realizowane w dniach od 01 do 24 września 2023 roku. Zakres prac obejmował ocenę bezpieczeństwa aplikacji OWASP Juice Shop przetwarzającej dane osobowe, na co składało się przeprowadzenie następujących zadań:

- Testy bezpieczeństwa

Ocena została przeprowadzona zgodnie z rynkowymi standardami weryfikowania bezpieczeństwa systemów informatycznych opracowanymi przez OWASP, w tym głównie OWASP Application Security Verification Standard v4 (L1) obejmujący również podatności z listy OWASP Top 10 2021.

Ocena została wykonana podejściem białej skrzynki (white-box).

Ocena została przeprowadzona przez zespół Gabrieli Czarneckiej, w którego skład wchodził:

- Gabriela Czarnecka

Lista znalezionych problemów:

#	Nazwa	Krytyczność	Status
1	Możliwość zalogowania się na konto administratora	P1- CRITICAL	Otwarta
2	Możliwość przejęcia konta użytkownika poprzez uzyskanie ciasteczek sesyjnych	P2- HIGH	Otwarta
3	Możliwość podglądania koszyków innych użytkowników	P3- MEDIUM	Otwarta
4	Możliwość modyfikowania zawartości koszyków innych użytkowników	P3- MEDIUM	Otwarta
5	Możliwość uzyskania dostępu do zasobów serwera aplikacji	P3- MEDIUM	Otwarta
6	Nieautoryzowany dostęp do katalogu ftp	P3- MEDIUM	Otwarta
7	Słaba implementacja mechanizmu przypominania hasła	P5- INFORMATIONAL	Otwarta
8	Słaba polityka haseł	P5- INFORMATIONAL	Otwarta
9	Komunikat umożliwiający enumerację	P5-	Otwarta



#	Nazwa	Krytyczność	Status
	użytkowników	INFORMATIONAL	
10	Wykorzystanie biblioteki z krytycznymi podatnościami	P5- INFORMATIONAL	Otwarta

Opis statusów:

- **Otwarta** – podatność obecna w systemie lub aplikacji.
- **Zamknięta** – podatność, która została wyprowadzona całkowicie.
- **Zmitygowana** – podatność, która została wyprowadzona częściowo co efektywnie wpływa na zmniejszenie ryzyka z nią związanego.

5. Rezultaty prac

Ocena podatności systemów na zagrożenia została zobrazowana za pomocą następujących miar wpływu podatności na bezpieczeństwo danych lub infrastruktury w skali (na podstawie VRT- Bugcrowd's Vulnerability Rating Taxonomy) :

Krytyczność	Opis
P1- CRITICAL	Podatności, które powodują eskalację uprawnień na platformie z nieuprzywilejowanych do administratora, umożliwiają zdalne wykonanie kodu, kradzież środków finansowych itp. Przykłady: luki, które powodują zdalne wykonanie kodu, takie jak obejście uwierzytelniania pionowego, SSRF, XXE, SQL Injection, obejście uwierzytelniania użytkownika.
P2- HIGH	Podatności w zabezpieczeniach, które wpływają na bezpieczeństwo platformy, w tym na obsługiwane przez nią procesy. Przykłady: boczne obejście uwierzytelniania, Stored XSS, niektóre CSRF w zależności od wpływu.
P3- MEDIUM	Podatności w zabezpieczeniach, które mają wpływ na wielu użytkowników i wymagają niewielkiej lub żadnej interakcji użytkownika do uruchomienia. Przykłady: Reflective XSS, Direct object reference, URL Redirect, niektóre CSRF w zależności od wpływu.
P4- LOW	Problemy, które mają wpływ na pojedynczych użytkowników i wymagają interakcji lub znaczących warunków wstępnych (MitM) do uruchomienia. Przykłady: typowe błędy, informacje o debugowaniu, zawartość mieszana.
P5- INFORMATIONAL	Słabości, których nie da się wykorzystać i luki, których nie da się naprawić. Przykłady: Najlepsze praktyki, środki łagodzące, kwestie, które wynikają z projektu lub akceptowalnego ryzyka biznesowego dla klienta, takie jak stosowanie CAPTCHA.

W kolejnych podrozdziałach prezentujemy szczegółowe wyniki przeprowadzonych testów bezpieczeństwa.

(P1- CRITICAL) 5.1 Możliwość zalogowania się na konto administratora

Krytyczność	P1- CRITICAL
Status	Otwarta

Podsumowanie

Punkt końcowy aplikacji: **rest/user/login** jest podatny na SQL Injection, co umożliwia **zalogowanie się do aplikacji na konto administratora**.

Więcej informacji:

- https://owasp.org/Top10/A03_2021-Injection/
- <https://owasp.org/www-project-proactive-controls/v3/en/c3-secure-database>
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x13-V5-Validation-Sanitization-Encoding.md>
- <https://cwe.mitre.org/data/definitions/89.html>

Warunki niezbędne do wykorzystania podatności

Brak.

Szczegóły techniczne

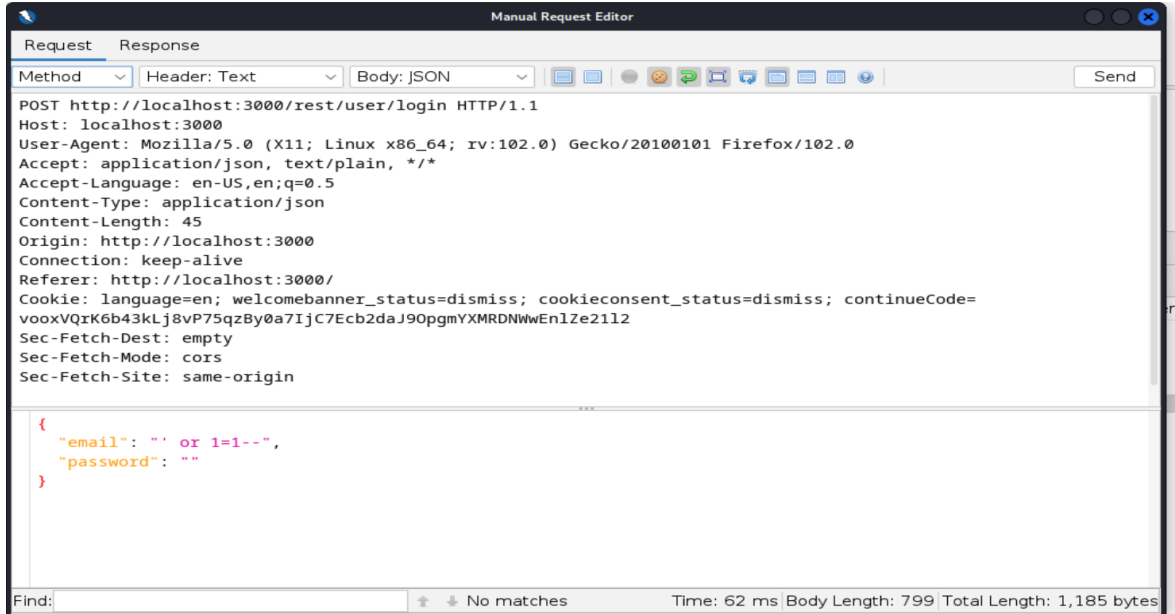
Po wysłaniu requestu do punktu końcowego aplikacji: **rest/user/login** z następującym payloadem:

```
{
  "email": "' or 1=1 - '",
  "password": ""
}
```

gdzie:

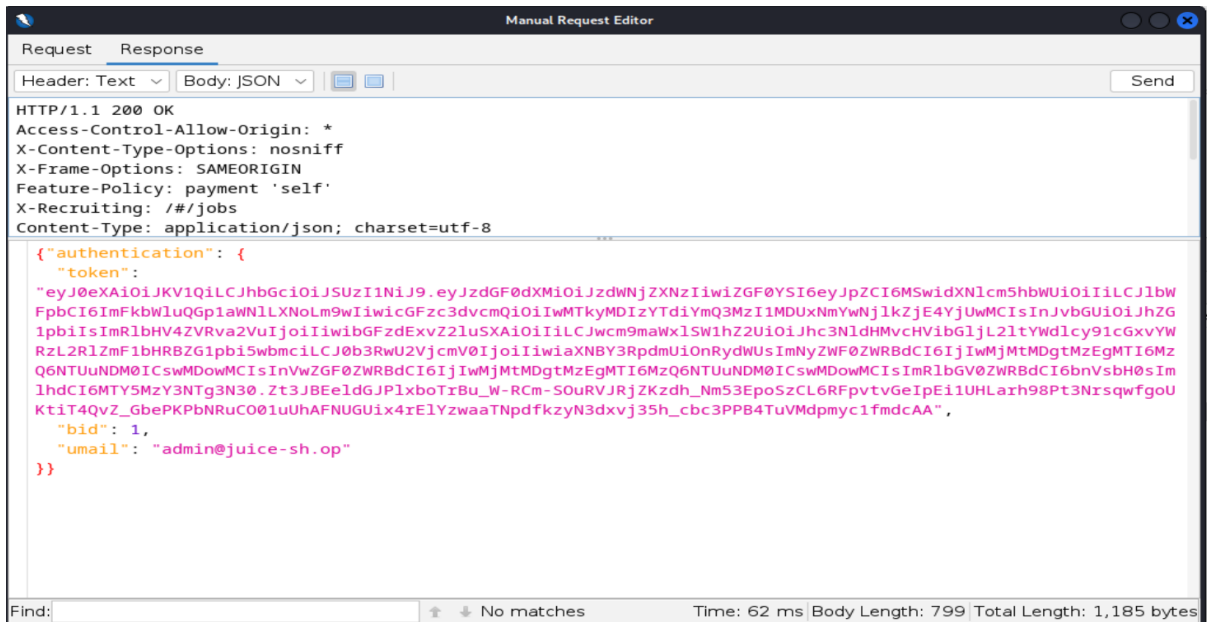
1. Znak ' zamyka nawiasy w zapytaniu SQL.
2. 'OR' w zapytaniu SQL zwróci wartość true, jeśli którakolwiek z jego stron jest prawdziwa. Ponieważ **1=1** jest zawsze prawdziwe, cała instrukcja jest prawdziwa. W ten sposób zostaje przekazana informacja serwerowi, że e-mail jest ważny i zaloguje nas do użytkownika o pierwszym napotkanym identyfikatorze, który przeważnie jest kontem administratora.
3. Znak - jest używany w SQL do komentowania danych, wszelkie ograniczenia dotyczące logowania nie będą już działać, ponieważ są interpretowane jako komentarz.

Request do punkt końcowego rest/user/login



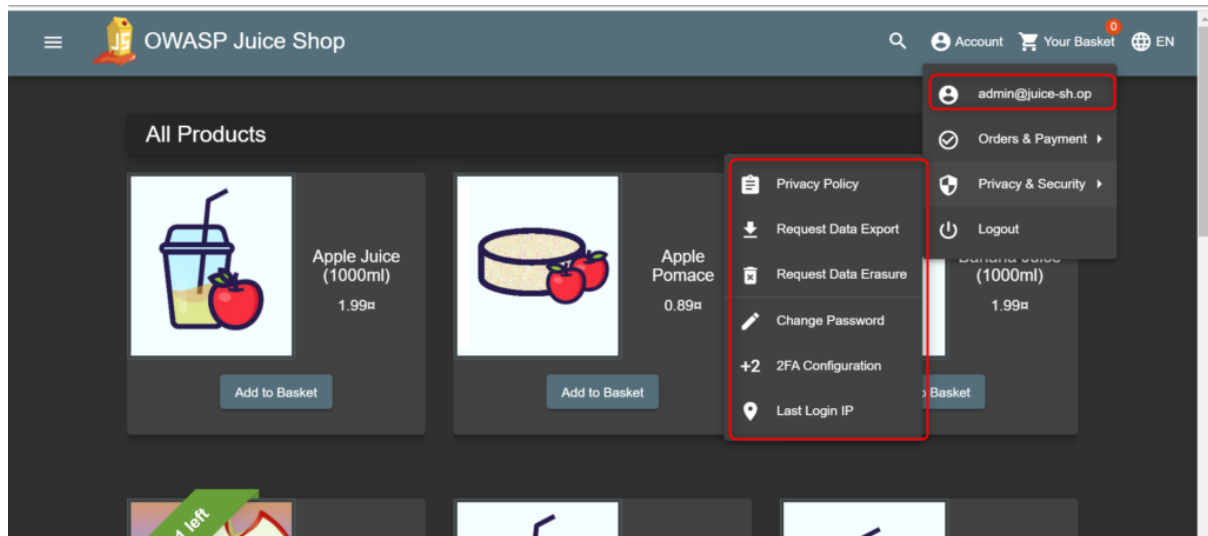
Otrzymuje odpowiedź świadczącą o prawidłowej autentykacji do aplikacji użytkownika z id=1 i adresem e-mail = admin@juice-shop.op

Odpowiedź świadcząca o pomyślnej autentykacji



Fakt ten potwierdzają dane wyświetlane bezpośrednio w aplikacji Juice Shop.

Juice Shop po poprawnej autentykacji na punkt końcowy `rest/user/login`



Lokalizacja problemu

Problem znajduje się w punkcie końcowym `rest/user/login`

Rekomendacje

Zaleca się wprowadzenie mechanizmu polegającego na **walidowaniu danych wejściowych** z wykorzystaniem listy dozwolonych wyrażeń (tzw. "Allow- list"), co powinno wyeliminować wykonanie zapytania zawierającego wartości, które nie występują na predefiniowanej liście.

Kolejną opcją jest użycie **przygotowanych instrukcji** (ang. "Prepared statements"), których użycie zapewnia, że atakujący nie jest w stanie zmienić intencji zapytania, nawet jeśli polecenia SQL są wstawiane przez atakującego.

Inną możliwą opcją przeciwdziałania atakom SQL Injection są **procedury przechowywane** (ang. "Stored procedures"), które polegają na przechowywaniu procedur kodu SQL w bazie danych.

Ostatnią metodą może być **filtrowanie i stosowanie znaków ucieczki w ciągach wprowadzanych przez użytkownika**. Technika ta ma na celu uniknięcie wprowadzania danych przez użytkownika przed umieszczeniem ich w zapytaniu. Technika ta powinna być stosowana tylko w ostateczności, gdy żadne z powyższych rozwiązań nie jest możliwe.

Więcej informacji:

- <https://owasp.org/www-project-proactive-controls/v3/en/c3-secure-database/>
- https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html
- https://owasp.org/Top10/A03_2021-Injection/

(P2- HIGH) 5.2 Możliwość przejęcia konta użytkownika poprzez uzyskanie ciasteczek sesyjnych

Krytyczność	P2- HIGH
Status	Otwarta

Podsumowanie

Wstrzyknięcie kodu JavaScript w wyszukiwarkę produktów powoduje jego wykonanie po stronie przeglądarki aplikacji (atakującemu zwracana jest zawartość ciasteczka z aktywną sesją użytkownika). Podatność może doprowadzić do **przejęcia konta użytkownika** – tymczasowego lub stałego jeżeli atakującemu uda się zmienić ustawienia konta .

Więcej informacji:

- https://owasp.org/Top10/A03_2021-Injection/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x13-V5-Validation-Sanitization-Encoding.md>
- <https://cwe.mitre.org/data/definitions/79.html>

Warunki niezbędne do wykorzystania podatności

1. Atakujący musi użyć socjotechniki (tj. Social Engineering) żeby ofiara weszła na stronę z użyciem spreparowanego linka zawierającego payload z kodem JavaScript.
2. Ofiara musi kliknąć w spreparowanego linka i dodatkowo posiadać aktywną sesję w aplikacji.

Szczegóły techniczne

Po zalogowaniu do aplikacji wprowadzono następujący kod JavaScript do inputu, który służy do wyszukiwania produktów :

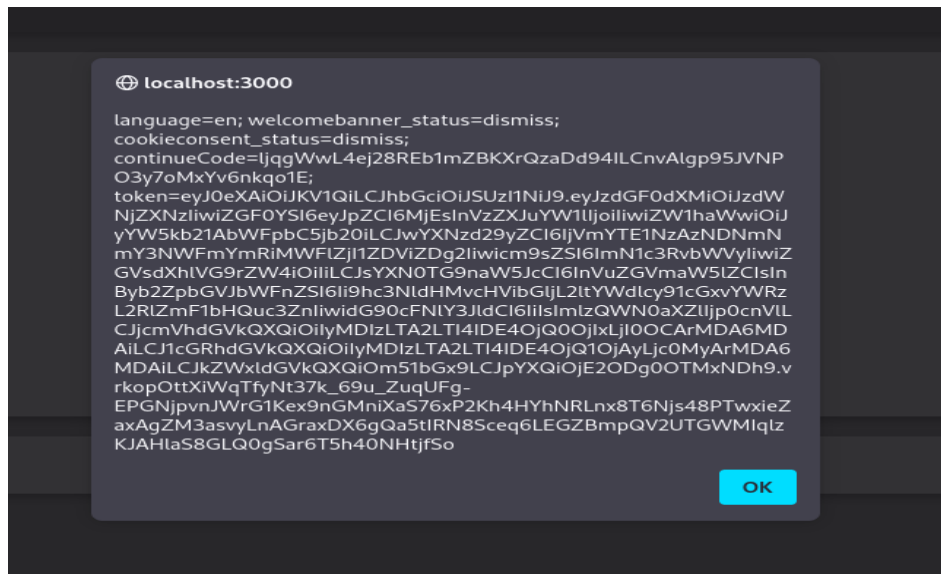
```
<iframe width='1' height='1' src='javascript:alert(document.cookie)'>
```

Umieszczenie inputu do wyszukiwania produktów



Wprowadzony do inputu kod JavaScript wykonuje się w przeglądarce aplikacji i zwracana jest zawartość ciasteczek sesyjnych.

Zawartość ciasteczek sesyjnych po wykonaniu kodu JavaScript



Lokalizacja problemu

Problem znajduje się w inputcie umiejscowionym na stronie głównej sklepu, służącym do wyszukiwania produktów.

Rekomendacje

Zaleca się usprawnić mechanizm odpowiedzialny za walidowanie i sanityzowanie (eskejpowane i enkodowanie, czyli podmianę znaków specjalnych na odpowiednie znaki kontrolne, które dadzą informacje interpreterowi żeby ich nie wykonywał) danych wejściowych po stronie przeglądarki internetowej.

Więcej informacji:

- [https://cheatsheetseries.owasp.org/cheatsheets/Cross Site Scripting Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/Cross_Site_Scripting_Prevention_Cheat_Sheet.html)
- [https://cheatsheetseries.owasp.org/cheatsheets/DOM based XSS Prevention Cheat Sheet.html](https://cheatsheetseries.owasp.org/cheatsheets/DOM_based_XSS_Prevention_Cheat_Sheet.html)
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x13-V5-Validation-Sanitization-Encoding.md>
- https://owasp.org/Top10/A03_2021-Injection/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x12-V3-Session-management.md>

(P3- MEDIUM) 5.3 Możliwość podglądania koszyków innych użytkowników

Krytyczność	P3- MEDIUM
Status	Otwarta

Podsumowanie

Brak poprawnej implementacji w punkcie końcowym `/rest/basket/{basket_ID}` umożliwia każdemu użytkownikowi, posiadającemu konto w sklepie na **odczytanie zawartości koszyka innego użytkownika**.

Więcej informacji:

- https://owasp.org/www-community/Broken_Access_Control
- <https://cwe.mitre.org/data/definitions/284.html>

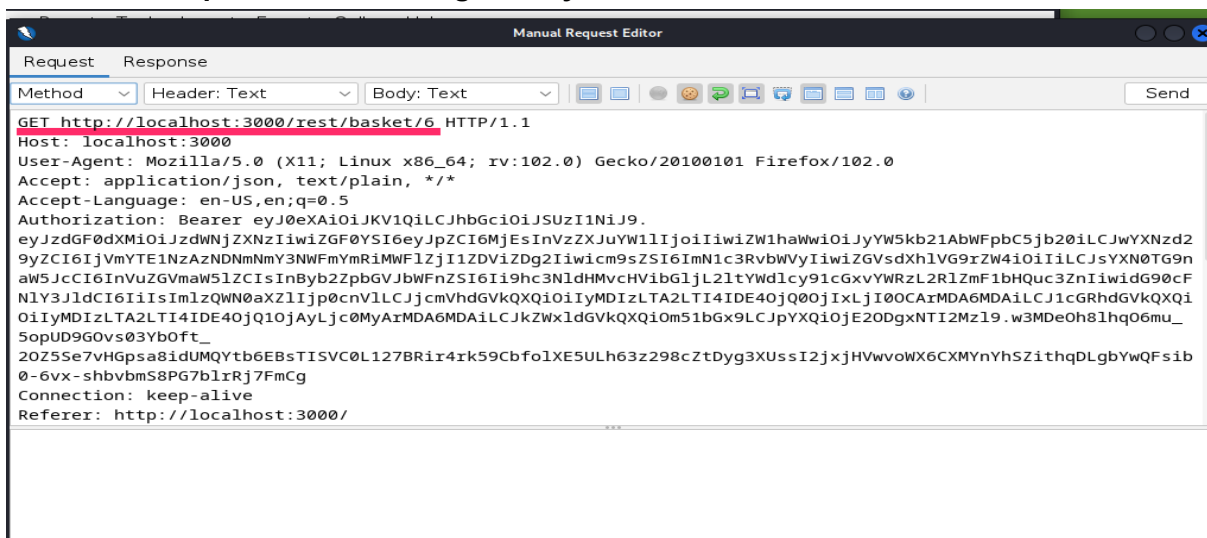
Warunki niezbędne do wykorzystania podatności

Posiadanie konta w aplikacji Juice Shop.

Szczegóły techniczne

Po założeniu konta w sklepie i zalogowaniu do aplikacji przez ww. punkt końcowy istnieje możliwość podejrzenia zawartości swojego koszyka (w moim przypadku koszyk posiada **id = 6**).

Zawartość Request'u dla własnego koszyka





Zawartość Response dla własnego koszyka

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *

{
  "status": "success",
  "data": {
    "id": 6,
    "coupon": null,
    "UserId": 21,
    "createdAt": "2023-06-28T18:44:38.243Z",
    "updatedAt": "2023-06-28T18:44:38.243Z",
    "Products": [
      {
        "id": 6,
        "name": "Banana Juice (1000ml)",
        "description": "Monkeys love it the most.",
        "price": 1.99,
        "deluxePrice": 1.99,
        "image": "banana_juice.jpg",
        "createdAt": "2023-06-28T18:41:29.030Z",
        "updatedAt": "2023-06-28T18:41:29.030Z",
        "deletedAt": null,
        "BasketItem": {
          "productId": 6
        }
      }
    ]
  }
}
```

Na tej podstawie można domniemywać, że id koszyków tworzone jest sekwencyjnie (1... n). Będąc zalogowanym wciąż jako ten sam użytkownik i zmieniając id koszyka na inny (w moim przypadku na id = 7) punkt końcowy `/rest/basket/{basket_ID}` zwraca kod odpowiedzi 200 oraz zawartość koszyka innego użytkownika.

Zawartość Response dla koszyka innego użytkownika

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *

{
  "status": "success",
  "data": {
    "id": 7,
    "coupon": null,
    "UserId": 24,
    "createdAt": "2023-06-30T19:15:16.265Z",
    "updatedAt": "2023-06-30T19:15:16.265Z",
    "Products": [
      {
        "id": 6,
        "name": "Banana Juice (1000ml)",
        "description": "Monkeys love it the most.",
        "price": 1.99,
        "deluxePrice": 1.99,
        "image": "banana_juice.jpg",
        "createdAt": "2023-06-28T18:41:29.030Z",
        "updatedAt": "2023-06-28T18:41:29.030Z",
        "deletedAt": null,
        "BasketItem": {
          "productId": 6
        }
      }
    ]
  }
}
```



W celu potwierdzenia powyższego został uruchomiony skany automatyczny dla koszyków o id w przedziale od 1 do 25.

Skan automatyczny dla koszyków o id 1-25

Task ID ^	Message Type	Code	Reason	RTT	Size Resp. Header	Size Resp. Body	Highest Alert	State	Payloads
0	Original	200 OK		175 ms	386 bytes	530 bytes	Medium		
1	Fuzzed	200 OK		346 ms	387 bytes	1,310 bytes	Reflected		1
2	Fuzzed	200 OK		685 ms	386 bytes	557 bytes	Reflected		2
3	Fuzzed	200 OK		692 ms	386 bytes	557 bytes	Reflected		3
4	Fuzzed	200 OK		529 ms	386 bytes	558 bytes	Reflected		4
5	Fuzzed	200 OK		668 ms	386 bytes	946 bytes	Reflected		5
6	Fuzzed	200 OK		379 ms	386 bytes	530 bytes	Reflected		6
7	Fuzzed	200 OK		437 ms	386 bytes	530 bytes			7
8	Fuzzed	200 OK		773 ms	384 bytes	30 bytes			8
9	Fuzzed	200 OK		435 ms	384 bytes	30 bytes			9
10	Fuzzed	200 OK		510 ms	384 bytes	30 bytes			10
11	Fuzzed	200 OK		1.06 s	384 bytes	30 bytes			11
12	Fuzzed	200 OK		472 ms	384 bytes	30 bytes			12
13	Fuzzed	200 OK		743 ms	384 bytes	30 bytes			13
14	Fuzzed	200 OK		876 ms	384 bytes	30 bytes			14
15	Fuzzed	200 OK		1.19 s	384 bytes	30 bytes			15
16	Fuzzed	200 OK		853 ms	384 bytes	30 bytes			16
17	Fuzzed	200 OK		1.02 s	384 bytes	30 bytes			17

Można zauważyć, że rozmiar koszyków o id od 1 do 7 jest większy niż rozmiar koszyków o id od 17 do 25. Może to wskazywać na brak artykułów w koszyku lub brak takich koszyków. Niemniej jednak w każdym przypadku punkt końcowy zwracał kod odpowiedzi 200, co potwierdza, że aplikacja nie posiada mechanizmu weryfikującego właściciela danego koszyka i dowolny użytkownik może podejrzeć zawartość koszyka innego użytkownika.

Lokalizacja problemu

Problem znajduje się w punkcie końcowym `/rest/basket/{basket_ID}`.

Rekomendacje

Zaleca się usprawnić mechanizm odpowiedzialny za weryfikację dostępu do zasobów. Użytkownik powinien mieć dostęp tylko do tych zasobów, których jest właścicielem. Wskazane jest korzystanie z jednego centralnego modułu autoryzacji i zaimplementowanie aplikacji tak, aby przechodziły przez niego wszelkie operacje wykonywane w aplikacji.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Testing_Automation_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- <https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls>
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/

(P3- MEDIUM) 5.4 Możliwość modyfikowania zawartości koszyków innych użytkowników

Krytyczność	P3- MEDIUM
Status	Otwarta

Podsumowanie

Brak poprawnej implementacji w punkcie końcowym **/api/BasketItems** umożliwia każdemu użytkownikowi, posiadającemu założone konto w sklepie na **zmianę zawartości koszyka innego użytkownika**.

Więcej informacji:

- https://owasp.org/www-community/Broken_Access_Control
- <https://cwe.mitre.org/data/definitions/284.html>

Warunki niezbędne do wykorzystania podatności

Posiadanie konta w aplikacji Juice Shop.

Szczegóły techniczne

Po poprawnej autentykacji do aplikacji Juice Shop mam możliwość dodawania artykułów do własnego koszyka poprzez punkt końcowy **/api/BasketItems**.

W przypadku, gdy za pośrednictwem tego endpoint'a próbuje dodać więcej niż jeden produkt do koszyka **użytkownika, dla którego dokonałam autentykacji** aplikacja pozwala na zapisanie tylko drugiego produktu.

W przypadku, gdy wyśle żądanie pod ten sam punkt końcowy, ale z innym payload'em, tj. z:

1. id produktu i ilością, która odnosi się do koszyka **użytkownika, dla którego dokonałam autentykacji** (w moim przypadku był to użytkownik o id = 6),
2. id produktu i ilością, która odnosi się do koszyka **innego, drugiego użytkownika** (w moim przypadku użytkownika o id = 7),

to zwracana jest odpowiedź świadcząca o poprawnym dodaniu produktu do koszyka drugiego użytkownika (użytkownika z id = 7).

The screenshot shows the 'Manual Request Editor' window. The 'Response' tab is selected. The 'Header' dropdown is set to 'Text' and the 'Body' dropdown is set to 'JSON'. The 'Send' button is visible. The response text is as follows:

```
HTTP/1.1 200 OK
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Content-Type: application/json; charset=utf-8
```

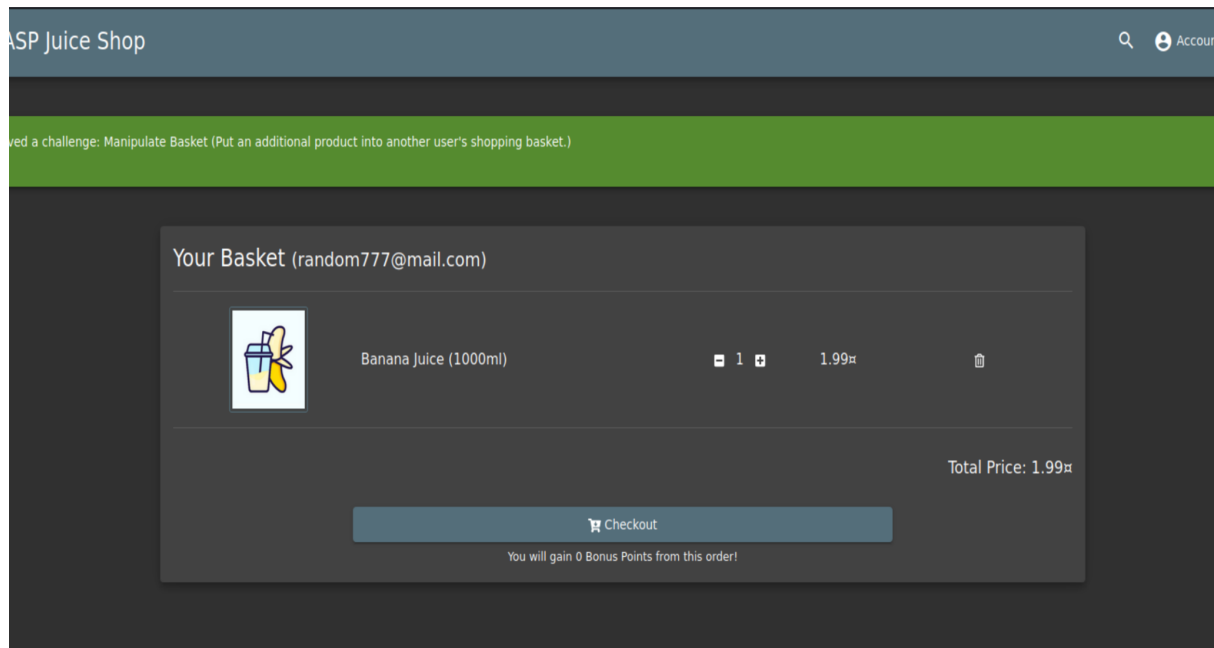
The body of the response is a JSON object:

```
{
  "status": "success",
  "data": {
    "id": 13,
    "ProductId": 6,
    "BasketId": "7",
    "quantity": 1,
    "updatedAt": "2023-09-05T16:36:24.216Z",
    "createdAt": "2023-09-05T16:36:24.216Z"
  }
}
```

At the bottom, the status bar shows 'Find: ' (empty), 'No matches', 'Time: 81 ms', 'Body Length: 157', and 'Total Length: 542 bytes'.

Strona 21

Zawartość koszyka użytkownika o id = 7



Lokalizacja problemu

Problem znajduje się w punkcie końcowym **/api/BasketItems**.

Rekomendacje

Zaleca się usprawnić mechanizm odpowiedzialny za weryfikację dostępu do zasobów. Użytkownik powinien móc wykonywać operacje tylko na tych zasobach, których jest właścicielem. Wskazane jest korzystanie z jednego centralnego modułu autoryzacji i zaimplementowanie aplikacji tak, aby przechodziły przez niego wszelkie operacje wykonywane w aplikacji.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Testing_Automation_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- <https://owasp.org/www-project-proactive-controls/v3/en/c7-enforce-access-controls>
- https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/05-Authorization_Testing/

(P3- MEDIUM) 5.5 Możliwość uzyskania dostępu do zasobów serwera aplikacji

Krytyczność	P3- MEDIUM
Status	Otwarta

Podsumowanie

Aplikacja umożliwia załadowanie awatara użytkownika z dowolnego adresu URL, co może umożliwić atakującemu **dostęp do zasobów znajdujących się na serwerze aplikacji** w sposób pośredni.

Więcej informacji:

- [https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_\(SSRF\)/](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/)
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x13-V5-Validation-Sanitization-Encoding.md>
- <https://cwe.mitre.org/data/definitions/918.html>

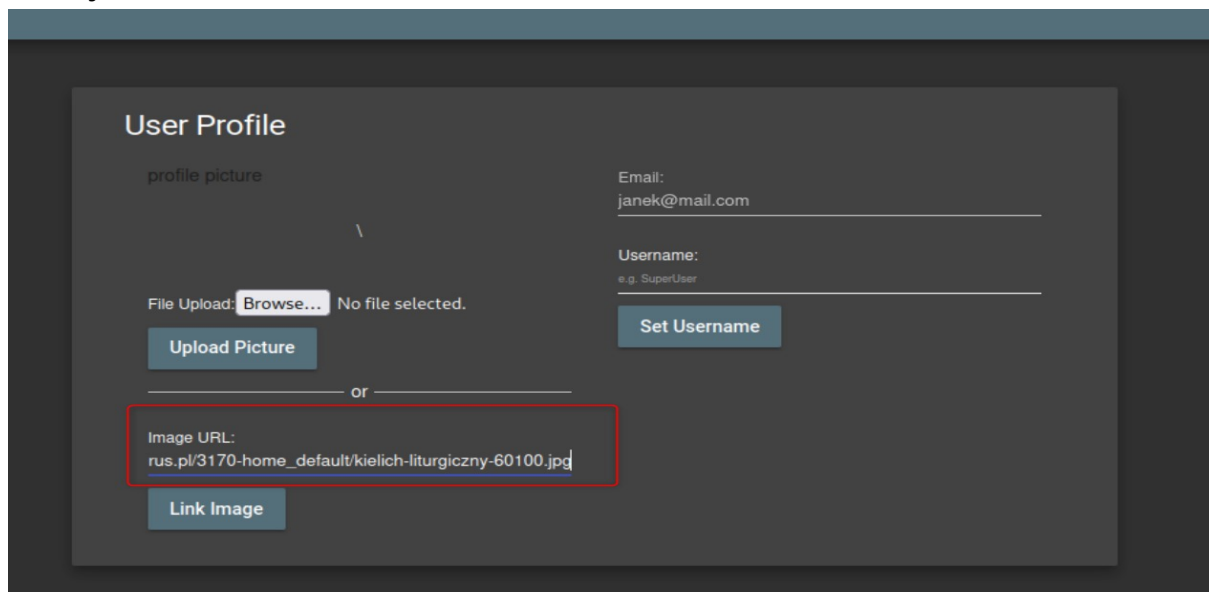
Warunki niezbędne do wykorzystania podatności

Posiadanie konta w aplikacji Juice Shop.

Szczegóły techniczne

Funkcjonalność dodawania awatara do profilu użytkownika umożliwia wstawienie dowolnego adresu URL.

Funkcjonalność dodawania awatara



User Profile

profile picture

Email: janek@mail.com

Username: e.g. SuperUser

File Upload: No file selected.

or

Image URL: rus.pl/3170-home_default/kielich-liturgiczny-60100.jpg



Po kliknięciu przycisku “Link image” serwer aplikacji wysyła request z podanym adresem URL, który jest podatny na SSRF.

Z kodu źródłowego aplikacji (plik **profileImageUrlUpload.ts**) wynika, że funkcja `profileImageUrlUpload()` sprawdza czy użytkownik jest zalogowany do aplikacji i jeśli tak, to wysyła request do podanego przez użytkownika URL’a. Ponieważ nie są zaimplementowane żadne kontrole wejściowe adresu URL, to poprzez manipulowanie tym parametrem atakujący może uzyskać dostęp do wewnętrznych zasobów serwera.

Kod źródłowy aplikacji - plik `profileImageUrlUpload.ts`

```
/*
 * Copyright (c) 2014-2023 Bjoern Kimminich & the OWASP Juice Shop contributors.
 * SPDX-License-Identifier: MIT
 */

import fs = require('fs')
import { Request, Response, NextFunction } from 'express'
import logger from '../lib/logger'

import { UserModel } from '../models/user'
import * as utils from '../lib/utils'
const security = require('../lib/insecurity')
const request = require('request')

module.exports = function profileImageUrlUpload () {
  return (req: Request, res: Response, next: NextFunction) => {
    if (req.body.imageUrl !== undefined) {
      const url = req.body.imageUrl
      if (url.match(/(.)*solve/challenges/server-side(.)*/) !== null) req.app.locals.abused_ssrf_bug = true
      const loggedInUser = security.authenticatedUsers.get(req.cookies.token)
      if (loggedInUser) {
        const imageRequest = request
          .get(url)
          .on('error', function (err: unknown) {
            UserModel.findByIdPk(loggedInUser.data.id).then(async (user: UserModel | null) => { return await
user?.update({ profileImage: url }) }).catch((error: Error) => { next(error) })
            logger.warn(`Error retrieving user profile image: ${utils.getErrorMessage(err)}; using image link directly`)
          })
          .on('response', function (res: Response) {
            if (res.statusCode === 200) {
              const ext = ['jpg', 'jpeg', 'png', 'svg', 'gif'].includes(url.split('.').slice(-1)[0].toLowerCase()) ?
url.split('.').slice(-1)[0].toLowerCase() : 'jpg'

              imageRequest.pipe(fs.createWriteStream(`frontend/dist/frontend/assets/public/images/uploads/${loggedInUser.d
ata.id}.${ext}`))
              UserModel.findByIdPk(loggedInUser.data.id).then(async (user: UserModel | null) => { return await
user?.update({ profileImage: `assets/public/images/uploads/${loggedInUser.data.id}.${ext}` }) }).catch((error:
Error) => { next(error) })
              } else UserModel.findByIdPk(loggedInUser.data.id).then(async (user: UserModel | null) => { return await
user?.update({ profileImage: url }) }).catch((error: Error) => { next(error) })
            }
          })
          .on('error', function (err: unknown) {
            UserModel.findByIdPk(loggedInUser.data.id).then(async (user: UserModel | null) => { return await
user?.update({ profileImage: url }) }).catch((error: Error) => { next(error) })
          })
      } else {
        next()
      }
    }
  }
}
```




```
        next(new Error('Blocked illegal activity by ' + req.socket.remoteAddress))
    }
}
res.location(process.env.BASE_PATH + '/profile')
res.redirect(process.env.BASE_PATH + '/profile')
}
}
```

Lokalizacja problemu

Problem znajduje się w funkcjonalności dodawania awatara w profilu użytkownika.

Rekomendacje

Zaleca się walidować i oczyszczać wszystkie dane jakie przychodzą od użytkownika, a także aplikacja powinna mieć możliwość wykonywania zapytań do sztywno określonych adresów URL w oparciu o tzw. "Allow-list".

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/assets/Server_Side_Request_Forgery_Prevention_Cheat_Sheet_SSRF_Bible.pdf
- [https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_\(SSRF\)/](https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_(SSRF)/)

(P3- MEDIUM) 5.6 Nieautoryzowany dostęp do katalogu ftp

Krytyczność	P3- MEDIUM
Status	Otwarta

Podsumowanie

Publiczny dostęp do katalogu ftp aplikacji (bez żadnych mechanizmów autoryzacji użytkowników).

Więcej informacji:

- https://owasp.org/Top10/A01_2021-Broken_Access_Control/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x12-V4-Access-Control.md>
- <https://cwe.mitre.org/data/definitions/285.html>

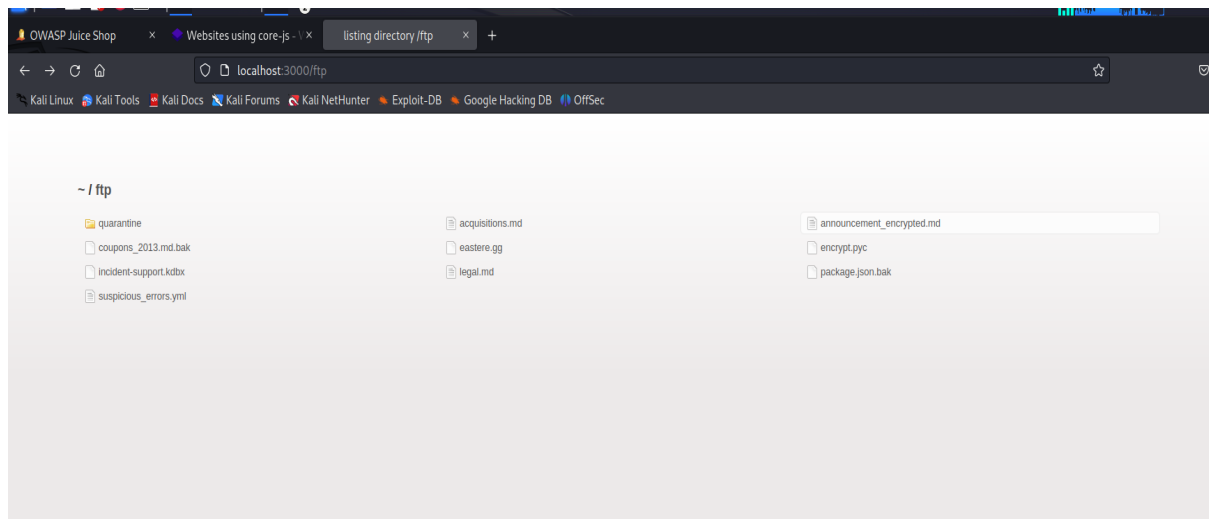
Warunki niezbędne do wykorzystania podatności

Brak.

Szczegóły techniczne

W wyniku uruchomionego skanu automatycznego aplikacji ujawniony został punkt końcowy **/ftp**. Dostęp do katalogu ftp nie jest w żaden sposób chroniony i możliwy do odczytania przez każdego użytkownika, który odkryje jego istnienie.

Zawartość serwera ftp



Lokalizacja problemu

Problem znajduje się w punkcie końcowym **/ftp**



Rekomendacje

Należy zaimplementować mechanizm odpowiedzialny za autoryzację użytkowników, którzy powinni mieć dostęp do katalogu.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html
- https://owasp.org/Top10/A01_2021-Broken_Access_Control/

(P5- INFORMATIONAL) 5.7 Słaba implementacja mechanizmu przypominania hasła

Krytyczność	P5- INFORMATIONAL
Status	Otwarta

Podsumowanie

Brak poprawnej implementacji mechanizmu przypominania hasła przez użycie pytań bezpieczeństwa. Wiele pytań dotyczy często używanych lub łatwo dostępnych informacji, takich jak imię matki, nazwisko panięńskie matki, miejsce urodzenia itp. Te informacje mogą być znane lub łatwo zdobyte przez osoby trzecie, zwłaszcza za pomocą danych dostępnych publicznie w mediach społecznościowych lub poprzez poszukiwanie informacji w dokumentach lub innych publicznie dostępnych źródłach. W efekcie końcowym podatność może doprowadzić do **przejęcia konta użytkownika**.

Więcej informacji:

- <https://pages.nist.gov/800-63-3/sp800-63b.html>
- https://owasp.org/Top10/A04_2021-Insecure_Design/
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- <https://cwe.mitre.org/data/definitions/640.html>

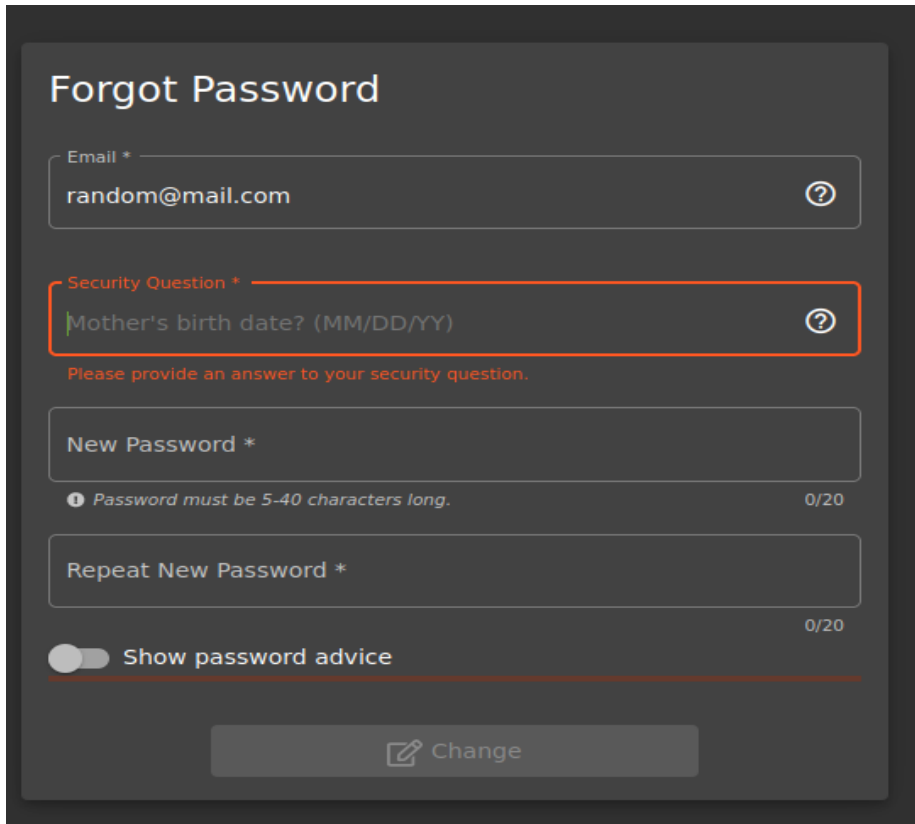
Warunki niezbędne do wykorzystania podatności

Atakujący musi znać adres e-mail ofiary.

Szczegóły techniczne

W funkcjonalności dotyczącej przypomnienia hasła (**Account > Login > Forgot your password?**) został zaimplementowany mechanizm polegający na możliwości zmiany hasła po udzieleniu odpowiedzi na pytanie bezpieczeństwa, które jest konfigurowane przy zakładaniu konta użytkownika.

Pytania bezpieczeństwa znajdujące się w mechanizmie przypomnienia hasła



Forgot Password

Email *
random@mail.com


Security Question *
Mother's birth date? (MM/DD/YY)

Please provide an answer to your security question.

New Password *
0/20
Password must be 5-40 characters long.

Repeat New Password *
0/20

☐ Show password advice

 Change

Lokalizacja problemu

Problem znajduje się w mechanizmie przypomnienia hasła.

Rekomendacje

Zaleca się zmianę mechanizmu przypominania hasła przy wykorzystaniu bardziej zaawansowanych metod autentykacji, takich jak np. wysyłanie linków do resetowania hasła na skonfigurowane wcześniej adresy e-mail.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Choosing_and_Using_Security_Questions_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://owasp.org/Top10/A04_2021-Insecure_Design/

(P5- INFORMATIONAL) 5.8 Słaba polityka haseł

Krytyczność	P5 - INFORMATIONAL
Status	Otwarta

Podsumowanie

Narzucanie konkretnych wymagań dot. złożoności hasła w mechanizmie rejestracji użytkownika. Wymuszanie konkretnej złożoności hasła, np. używania określonych typów znaków, może skłonić użytkowników do tworzenia przewidywalnych wzorców, takich jak "Hasło123!" lub "Imie1234". Takie wzorce są łatwe do odgadnięcia przez atakujących. W efekcie końcowym podatność może doprowadzić do **przejęcia konta użytkownika**.

Więcej informacji:

- <https://pages.nist.gov/800-63-3/sp800-63b.html>
- https://owasp.org/Top10/A04_2021-Insecure_Design/
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x11-V2-Authentication.md>
- <https://cwe.mitre.org/data/definitions/521.html>

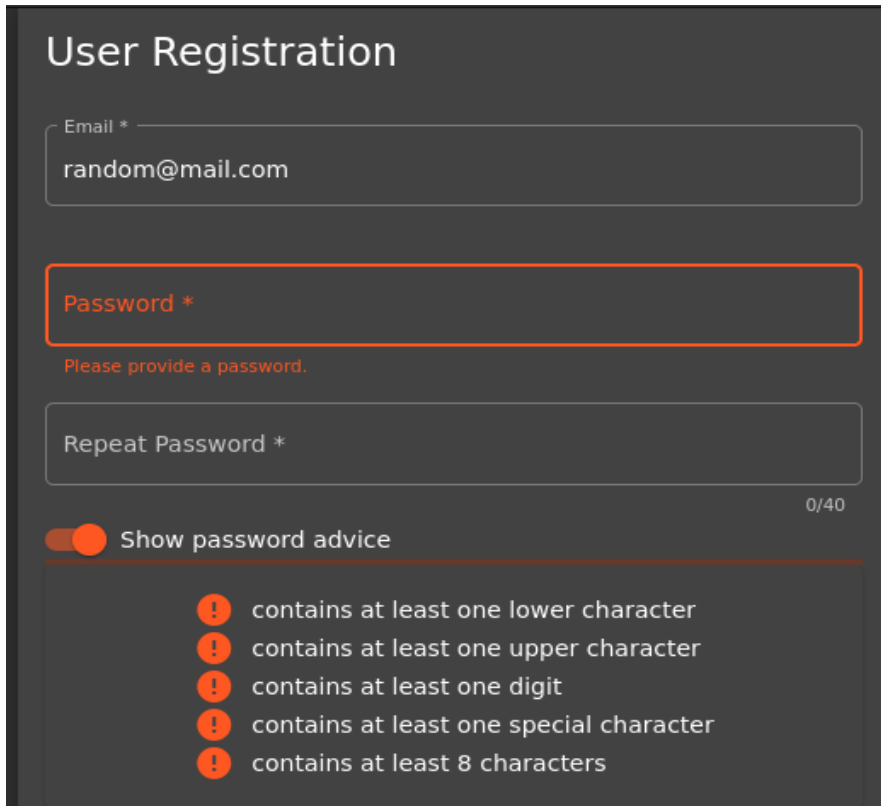
Warunki niezbędne do wykorzystania podatności

Atakujący musi znać adres e-mail ofiary.

Szczegóły techniczne

W funkcjonalności dotyczącej rejestracji użytkownika (**Account > Login > Not yet customer?**) został zaimplementowany mechanizm polegający na wymuszaniu wprowadzenia określonej złożoności hasła przez użytkownika.

Wymagania dotyczące złożoności hasła w mechanizmie rejestracji użytkownika



User Registration

Email *
random@mail.com

Password *

Please provide a password.

Repeat Password *

0/40

☒ Show password advice

- ! contains at least one lower character
- ! contains at least one upper character
- ! contains at least one digit
- ! contains at least one special character
- ! contains at least 8 characters

Lokalizacja problemu

Problem znajduje się w mechanizmie rejestracji użytkownika.

Rekomendacje

Zaleca się zniesienie wymogów dotyczących złożoności hasła takich jak: mała litera, duża litera, liczba, znak specjalny.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://owasp.org/Top10/A04_2021-Insecure_Design/
- <https://pages.nist.gov/800-63-3/sp800-63b.html>

(P5- INFORMATIONAL) 5.9 Komunikat umożliwiający enumerację użytkowników

Krytyczność	P5 - INFORMATIONAL
Status	Otwarta

Podsumowanie

W mechanizmie rejestracji użytkownika w przypadku, gdy próbuje się zarejestrować użytkownika na adres e-mail, który jest już przypisany do innego konta zwracany jest komunikat, że podany adres e-mail musi być unikalny. Taki komunikat daje atakującemu informacje o tym, że konto dla podanego adresu e-mail istnieje i może powodować próbę ataków, co w efekcie końcowym może doprowadzić do **przejęcia konta użytkownika**.

Więcej informacji:

- https://owasp.org/Top10/A04_2021-Insecure_Design/
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x11-V2-Authentication.md>
- <https://cwe.mitre.org/data/definitions/204.html>
- <https://pages.nist.gov/800-63-3/sp800-63b.html>

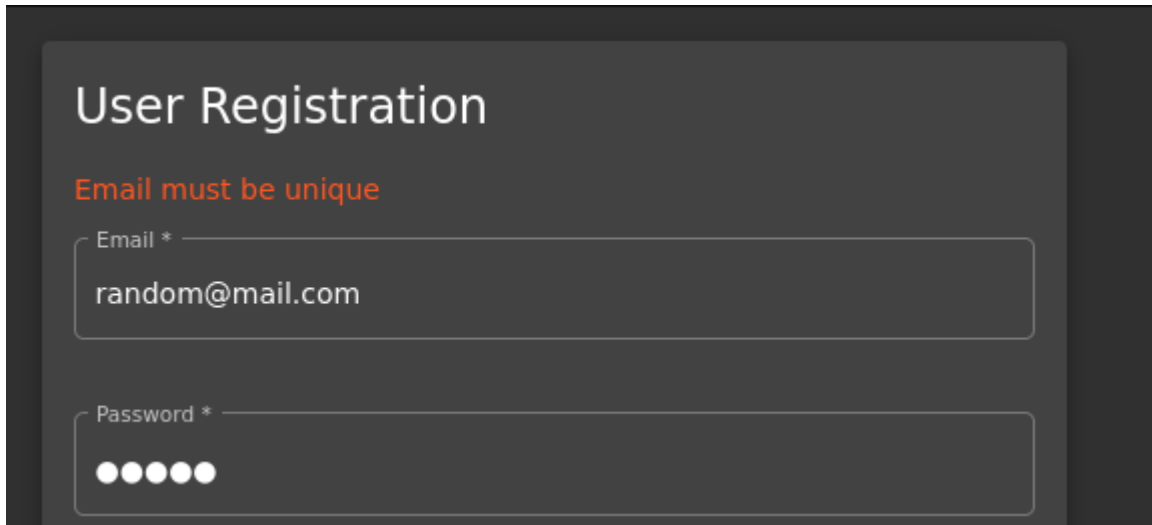
Warunki niezbędne do wykorzystania podatności

Złamanie hasła przypisanego do konta.

Szczegóły techniczne

W funkcjonalności dotyczącej rejestracji użytkownika (**Account > Login > Not yet customer?**) w przypadku, gdy próbuje się zarejestrować użytkownika na adres e-mail, który jest już przypisany do innego konta zwracany jest komunikat “Email must be unique”, co stanowi cenną informację dla atakującego, że podane konto istnieje.

Komunikat wyświetlany użytkownika przy próbie rejestracji na istniejące konto



Lokalizacja problemu

Problem znajduje się w mechanizmie rejestracji użytkownika.

Rekomendacje

Zaleca się zmianę komunikatu tak, by nie zawierał sugestii, że podane konto istnieje.

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html
- https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/
- https://owasp.org/Top10/A04_2021-Insecure_Design/
- <https://pages.nist.gov/800-63-3/sp800-63b.html>



(P5- INFORMATIONAL) 5.10 Wykorzystanie biblioteki z krytycznymi podatnościami

Krytyczność	P5- INFORMATIONAL
Status	Otwarta

Podsumowanie

Zaimplementowana wersja biblioteki sanitize-html posiada w swoich zależnościach bibliotekę lodash, która posiada kilka **krytycznych podatności**.

Więcej informacji:

- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/
- <https://github.com/OWASP/ASVS/blob/master/4.0/en/0x10-V1-Architecture.md>
- <https://cwe.mitre.org/data/definitions/1352.html>

Warunki niezbędne do wykorzystania podatności

Wykorzystanie biblioteki w funkcjonalności aplikacji.

Szczegóły techniczne

Zaimplementowana wersja biblioteki sanitize-html (1.4.2) w swoich zależnościach posiada bibliotekę lodash (wersja 4.17.20), która posiada podatności, takie jak:

- Regular Expression Denial of Service (ReDoS)
- Prototype Pollution
- Command Injection

Wersja biblioteki sanitize-html

```
161 "portscanner": "^2.2.0",
162 "prom-client": "^14.1.0",
163 "pug": "^3.0.0",
164 "replace": "^1.2.0",
165 "request": "^2.88.2",
166 "sanitize-filename": "^1.6.3",
167 "sanitize-html": "1.4.2",
168 "semver": "^7.3.2",
169 "sequelize": "^6.15.1",
170 "serve-index": "^1.9.1",
```



Podatności, które zawarte są w bibliotece lodash

```
lodash ≤ 4.17.20
Severity: critical
Regular Expression Denial of Service (ReDoS) in lodash - https://github.com/advisories/GHSA-x5rq-j2xg-h7qm
Prototype Pollution in lodash - https://github.com/advisories/GHSA-4xc9-xhrj-v574
Prototype Pollution in lodash - https://github.com/advisories/GHSA-fvqr-27wr-82fm
Prototype Pollution in lodash - https://github.com/advisories/GHSA-p6mc-m468-83gw
Command Injection in lodash - https://github.com/advisories/GHSA-35jh-r3h4-6jhm
Regular Expression Denial of Service (ReDoS) in lodash - https://github.com/advisories/GHSA-29mw-wpgm-hmr9
Prototype Pollution in lodash - https://github.com/advisories/GHSA-jf85-cpcp-j695
fix available via `npm audit fix --force`
Will install sanitize-html@2.11.0, which is a breaking change
node_modules/sanitize-html/node_modules/lodash
sanitize-html ≤ 2.3.1
Depends on vulnerable versions of lodash
node_modules/sanitize-html
```

Lokalizacja problemu

Problem znajduje się w bezpośrednio w bibliotece lodash, która wykorzystywana jest pośrednio w bibliotece sanitize-html.

Rekomendacje

Zaleca się podniesienie wersji biblioteki sanitize-html do najbardziej aktualnej wersji .

Więcej informacji:

- https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html
- https://cheatsheetseries.owasp.org/cheatsheets/Vulnerable_Dependency_Management_Cheat_Sheet.html
- https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/



INNE NIEPRAWIDŁOWOŚCI

W trakcie przeprowadzonych testów bezpieczeństwa stwierdzono również defekt w aplikacji - poprzez punkt końcowy **/api/Users** można utworzyć konto dla użytkownika, podając różną wartość w atrybucie password i passwordRepeat w body requestu. Walidacja, polegająca na sprawdzeniu czy wartości podane w tych polach są takie same, działa jedynie po stronie frontendu aplikacji.

Zawartość request'a wysłanego do punktu końcowego /api/Users

```
Request  Response
Method  Header: Text  Body: JSON  Send
POST http://localhost:3000/api/Users/ HTTP/1.1
Host: localhost:3000
User-Agent: Mozilla/5.0 (X11; Linux x86_64; rv:102.0) Gecko/20100101 Firefox/102.0
Accept: application/json, text/plain, */*
Accept-Language: en-US,en;q=0.5
Content-Type: application/json
Content-Length: 247
Origin: http://localhost:3000
Connection: keep-alive
Referer: http://localhost:3000/
Cookie: language=en; welcomebanner_status=dismiss; cookieconsent_status=dismiss; continueCode=73DnyDnKvZw7akVnQvIL4VYIlaIvCkF81c7MuRV0nr8P57eImv61004wMI Rn

{
  "email": "random@mail.com",
  "password": "password1!",
  "passwordRepeat": "password2!",
  "securityQuestion": {
    "id": 3,
    "question": "Mother's birth date? (MM/DD/YY)",
    "createdAt": "2023-09-21T16:36:39.683Z",
    "updatedAt": "2023-09-21T16:36:39.683Z"
  },
  "securityAnswer": "yes"
}
```

Odpowiedź serwera na wysłany request

```
Manual Request Editor
Request  Response
Header: Text  Body: JSON  Send
HTTP/1.1 201 Created
Access-Control-Allow-Origin: *
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Feature-Policy: payment 'self'
X-Recruiting: /#/jobs
Location: /api/Users/28
Content-Type: application/json; charset=utf-8
Content-Length: 312
ETag: W/"138-OV+Uw4dmqgjosFS0zPUyhPW1SZ8"
Vary: Accept-Encoding
Date: Sun, 16 Jul 2023 14:06:23 GMT
Connection: keep-alive
Keep-Alive: timeout=5

{
  "status": "success",
  "data": {
    "username": "",
    "role": "customer",
    "deluxeToken": "",
    "lastLoginIp": "0.0.0.0",
    "profileImage": "/assets/public/images/uploads/default.svg",
    "isActive": true,
    "id": 28,
    "email": "random88888@mail.com",
  }
}
```

Find: 400 Bad Request 6 ms 38 bytes Medium