Autor rozwiązania: Hubert Michalski - hm438596, 26 paź 2023

Do znalezienia wszystkich flag używałem jedynie:

- PyCharma, żeby wygodnie przeglądać kod
- Postmana, dużo wygodniej się modyfikuje requesty niż z terminala
- Pipedream RequestBin, do testowania XSS

1. FLAG{JeszczeJednaFlagaZaPunkty}

Na początku w całym projekcie wyszukałem sobie miejsc, gdzie mogą się dane flagi znajdować i zauważyłem, że jedna z nich została prawdopodobnie umieszczona w pliku “flag.txt” podczas tworzenia kontenera aplikacji.

```
cd /home/app/task/
python3 manage.py migrate
python3 manage.py loaddata --format=json - < fixture
rm fixture
echo FLAG{[redacted]} > /flag.txt
uwsgi --ini /home/app/app.ini --die-on-term

```

Zatem pozostało znaleźć podatność w aplikacji, która pozwoli na otworzenie danego pliku, wyszukując w projekcie komenty “open” ukazują się dwa miejsca:

```
def render_card(card):
    recipient = card.recipient
    content = card.content
    footer = card.sender.profile.footer
    with open(
        f"{os.path.dirname(os.path.realpath(__file__))}/card_templates/{card.template}",
        "r",
    ) as f:
        rendered = f.read()
```

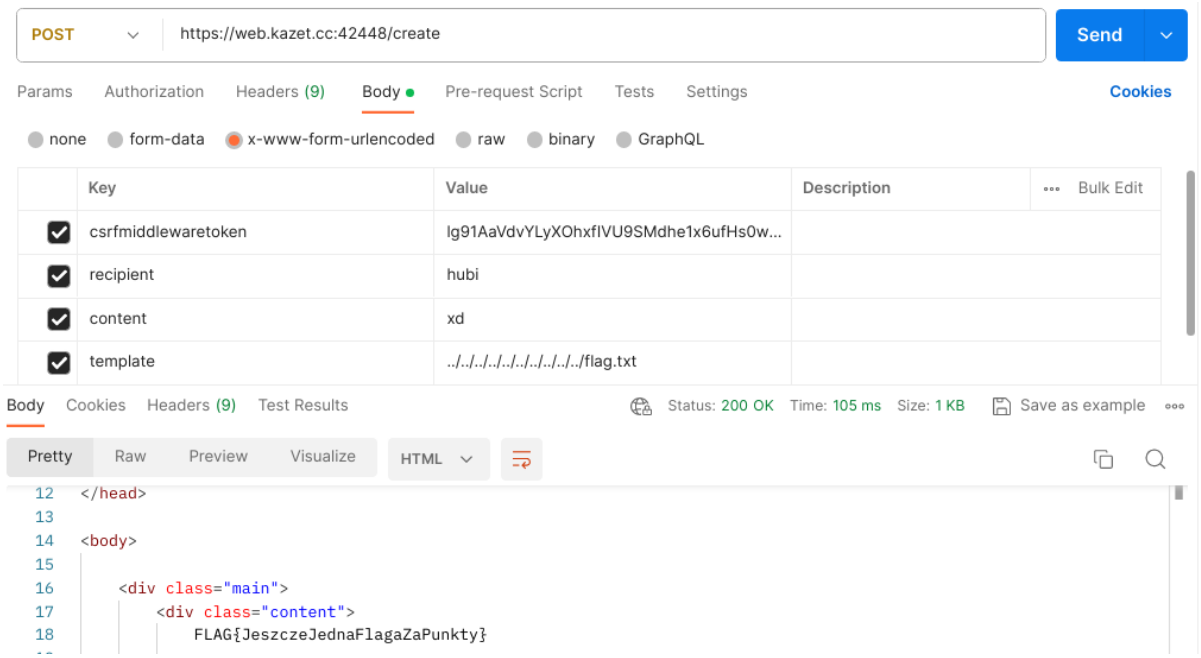
```
    def get(self, request, token):
        file_name = tempfile.mktemp(suffix=".png")
        subprocess.call(
            [
                "chromium",
                "--headless",
                "--disable-gpu",
                "--no-sandbox",
                "--screenshot=" + file_name,
                "http://127.0.0.1:3000/show-for-png/" + token,
            ]
        )
        with open(file_name, "rb") as f:
            return HttpResponse(f.read(), content_type="image/png")
```

Możemy jednak odrzucić od razu drugą opcję, ponieważ otwiera ona jedynie tymczasowy plik. W takim razie zainteresowanie pada na pierwszą opcję, widać, że

gdyby dało się przekazać nazwę pliku do “card.template” to uda się otworzyć plik który chcemy wykraść, na szczęście po sprawdzeniu jakiego typu jest to pole w modelu widzimy:

```
content = models.TextField()
template = models.TextField()
```

Co oznacza, że walidacja jest jedynie na stronie (gdzie przyjmuje tylko “coffee” i “normal”) ale gdy sami stworzymy requesta to możemy tam podać cokolwiek np. nazwę poszukiwanego pliku a raczej ścieżkę do niego. Wszystkie requesty są dostępne w tym folderze jako kolekcja Postmana i równoważne komendy CURL (jednak przyznaje się, że nie testowałem CURL, a jedynie używałem Postmana).



2. FLAG{71a4b4fd2214b808e4942dfb06c717878399a04c}

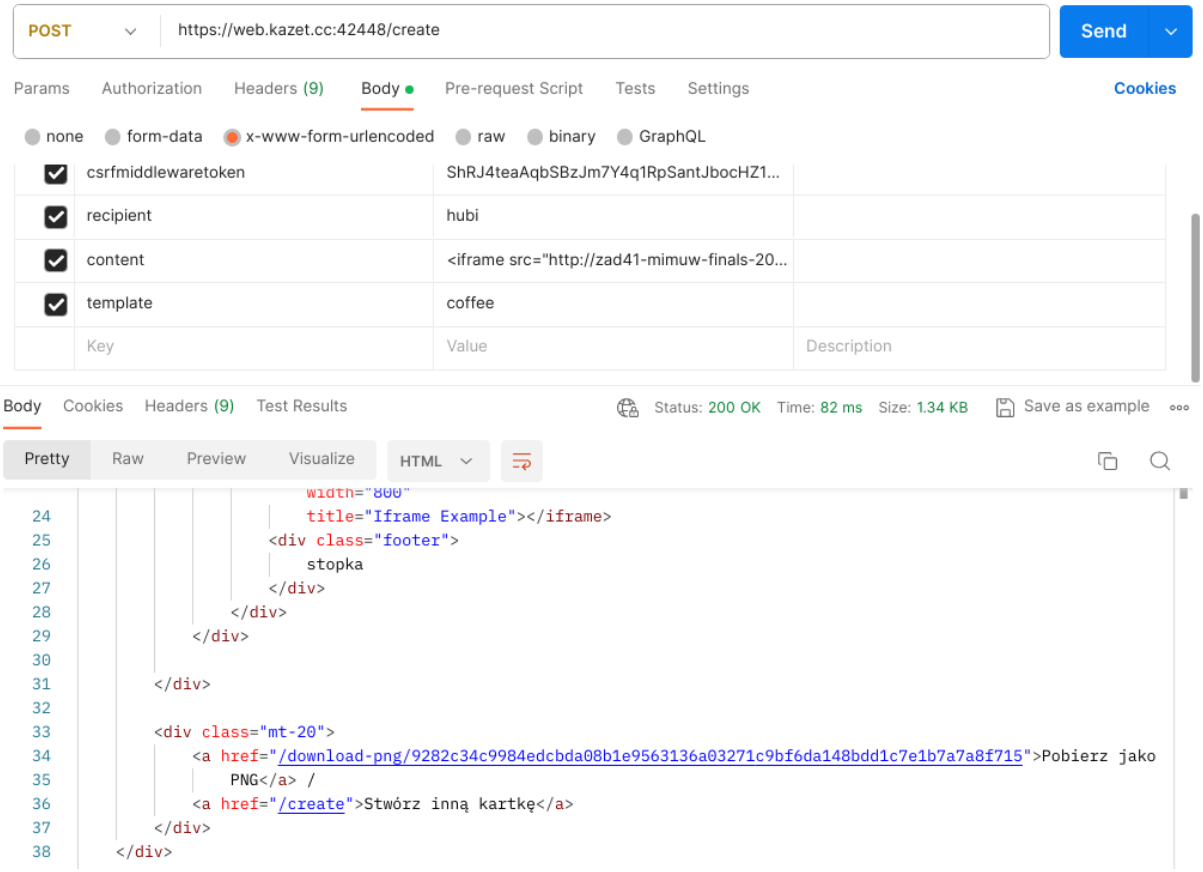
Kolejne miejsce, które wskazywałoby na to gdzie mogą znajdować się flagi jest to:

```
15  zad41-mimuw-finals-2023-super-secret-microservice:
16    volumes:
17      - ../zad41-mimuw-finals-2023.flag1:/static/index.html
18    build: super-secret-microservice/
19    restart: always
20    networks:
21      - network41-ext
```

Zatem gdyby udało się dostać do strony głównej danego mikroserwisu to prawdopodobnie dostalibyśmy flagę, ale serwis ten nie jest dostępny za pomocą publicznego IP a jedynie z sieci wewnętrznej naszych wszystkich mikroserwisów z pliku docker-compose. Od początku uwagę skupia kod który pominąłem przy poprzedniej fladze tj:

```
76  def get(self, request, token):
77      file_name = tempfile.mktemp(suffix=".png")
78      subprocess.call(
79          [
80              "chromium",
81              "--headless",
82              "--disable-gpu",
83              "--no-sandbox",
84              "--screenshot=" + file_name,
85              "http://127.0.0.1:3000/show-for-png/" + token,
86          ]
87      )
88      with open(file_name, "rb") as f:
89          return HttpResponse(f.read(), content_type="image/png")
```

Chciałem w jakiś sposób wywołać CURLa z tej maszyny do drugiego mikroserwisu, lecz szybko mi się przypomniało, że na zajęciach ta metoda była przedstawiona jako bezpieczna i ciężko coś z tym zrobić. Dalej jednak pomyślałem, że można wykorzystać fakt, że ta maszyna na chwile uruchamia przeglądarkę, żeby wykonać screenshota naszej kartki - na przykład jakoś przekierować się na stronę ”sekretnego” mikroserwisu, to się niestety zakończyło porażką. Ostatecznym pomysłem zostało osadzenie kodu strony tego mikroserwisu, tak aby był on widoczny na robionym przez maszynę screenie - aby zrealizować ten pomysł użyłem iframe i się szczęśliwie udało :) Przydała się także znajomość Dockera, że mikroserwisy które mają przypisany network potrafią rozwiązywać swoje nazwy.


POST
https://web.kazet.cc:42448/create
Send
Params
Authorization
Headers (9)
Body
Pre-request Script
Tests
Settings
Cookies
none
form-data
x-www-form-urlencoded
raw
binary
GraphQL
csrfmiddlewaretoken
ShRJ4teaAqbSBzJm7Y4q1RpSantJbocHZ1...
recipient
hubi
content
<iframe src="http://zad41-mimuw-finals-20...
template
coffee
Key
Value
Description
Body
Cookies
Headers (9)
Test Results
Status: 200 OK
Time: 82 ms
Size: 1.34 KB
Save as example
Pretty
Raw
Preview
Visualize
HTML
width="800"
title="Iframe Example"></iframe>

stopka

Pobierz jako
PNG /
Stwórz inną kartkę

3. FLAG{ToJestFlagaZeStopki}

Ostatnią flagę w kodzie możemy znaleźć w pliku “fixture”, który jest ładowany do bazy podczas skryptu startowego kontenera głównej aplikacji:

fixture app/app

```
            "user_permissions" : [],
            "username" : "admin"
        },
        "model" : "auth.user",
        "pk" : 1
    },
    {
        "fields" : {
            "footer" : "FLAG{[redacted]}",
            "user" : 1
        },
        "model" : "accounts.profile",
        "pk" : 1
    }
]
```

```
cd /home/app/task/
python3 manage.py migrate
python3 manage.py loaddata --format=json - < fixture
rm fixture
echo FLAG{[redacted]} > /flag.txt
uwsgi --ini /home/app/app.ini --die-on-term
```

Po chwili analizy kodu jesteśmy w stanie wywnioskować, że aby otrzymać daną flagę potrzebujemy dostać kartkę od admina, ponieważ każda kartka ma załączoną stopkę a więc w tym przypadku dostalibyśmy również poszukiwaną flagę. Po pierwsze widzimy, że nigdzie nie możemy wykorzystać SQL injection więc nie jesteśmy w stanie wykraść hasła admina, jednak w kodzie widzimy bota który odczytuje kartki admina. Domyślamy się zatem, że można wykorzystać XSS do wykradnięcia sesji (już w poprzednich flagach wykorzystaliśmy wstrzyknięcie iframe’a). Próba wykradnięcia ciasteczek jednak się nie może udać, ponieważ sessionid jest oznaczone jako HttpOnly, zatem trzeba szukać innych sposobów.

Name	Value	Dom...	Path	Expi...	Size	HttpOnly
sessionid	9535cp80pn54a5j8q6wb4pil4heyd2...	web....	/	202...	41	✓
csrftoken	hUx4TMjd7B88OhRkIsJtffjObqkChtcu	web....	/	202...	41	

Można jednak spróbować wysłać bezpośrednio kartkę z “urządzenia” admina, w tym celu wystarczy “jedynie” spreparować odpowiedniego requesta który miałby zostać wykonany po stronie admina, lecz nie jest to takie proste, ponieważ formularz został zabezpieczony przez csrf_token. Moim początkowym pomysłem było przesłanie do siebie samego csrfmiddlewaretoken a następnie napisanie fetch’a który ma wykonać admin z początkowo wyciągniętym tokenem, lecz w tym przypadku to nie działa ponieważ bot do każdej kartki loguje się osobno i csrftoken w ciasteczkach się zmienia więc admin dostawał cały czas 403. Rozwiązaniem tego problemu jest sklejenie obu tych kroków w jeden po stronie admina, zatem kod javascriptowy najpierw wykonywał ”wyciągnie” csrfmiddlewaretoken’a a następnie używał go w requescie który tworzył kartkę wysłaną do mnie. Kod przesyłanego JS’a jest ładnie sformowany w pliku z CURL’em :)