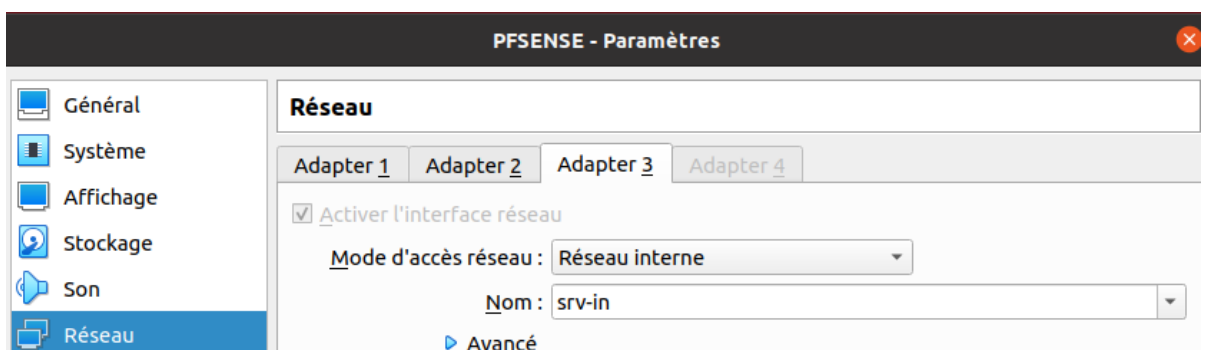
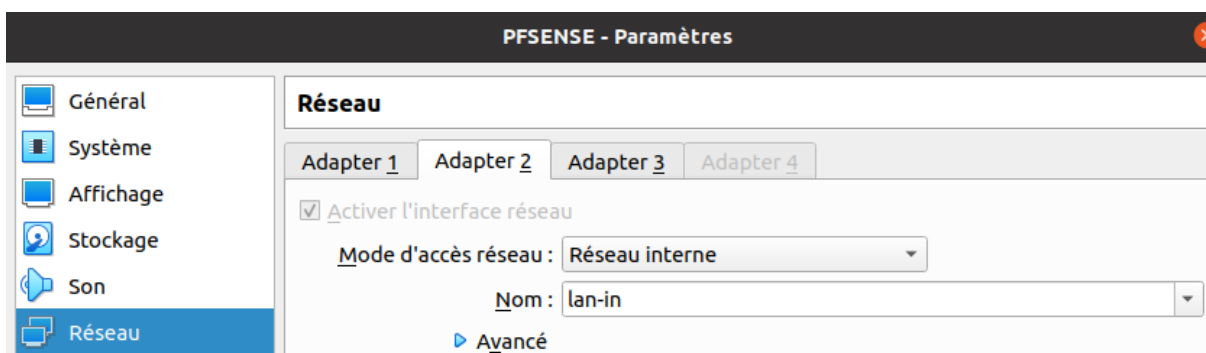
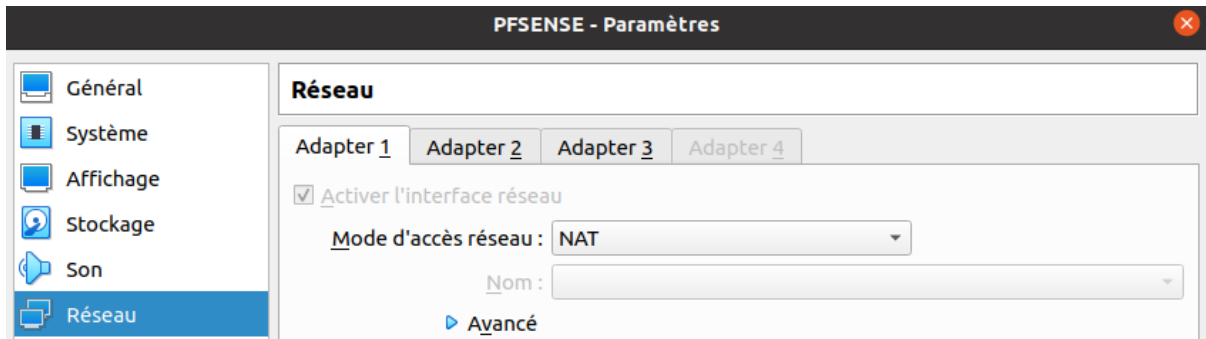


## TP01

### Manipulation : Préparer les machines

#### 1. Serveur PFSENSE

Vérification carte réseau:

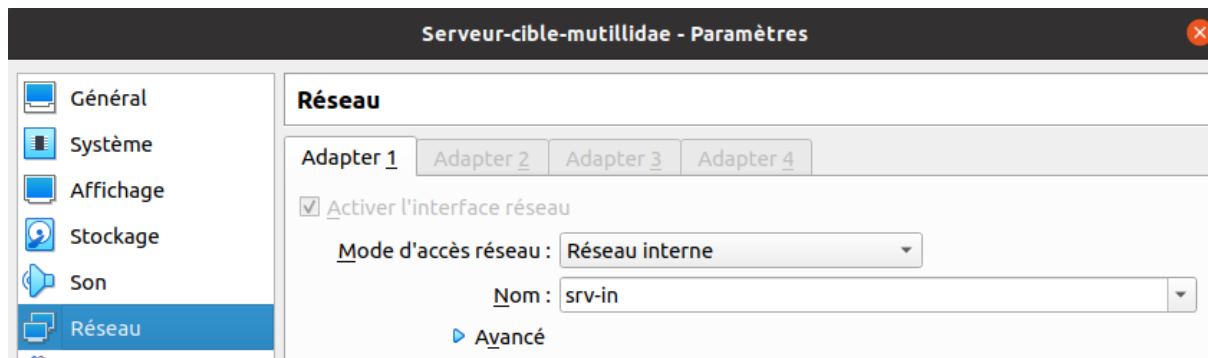


Voici les cartes réseau dans le serveur pfsense. On peut voir que la carte em0 est bien configurée en NAT. La carte em1 est configurée en réseau interne lan-in et la carte em2 est elle aussi en interne srv-in.

```
*** Welcome to pfSense 2.4.4-RELEASE-p1 (amd64) on pfSense ***  
  
WAN (wan)      -> em0      ->  
LAN_IN (lan)   -> em1      -> v4: 192.168.50.254/24  
SRV_IN (opt1) -> em2      -> v4: 172.16.10.254/24
```

## 2. Serveur cible-mutillidae

Vérification carte réseau:

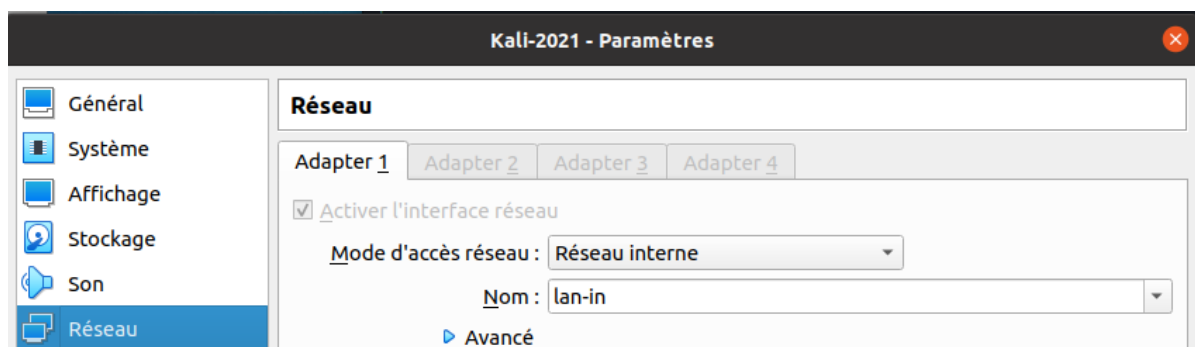


Notre carte réseau a la bonne adresse IP. 172.16.10.5 Elle est donc bien dans le réseau srv-in.

```
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_
0
    link/ether 08:00:27:cb:2d:d0 brd ff:ff:ff:ff:ff:ff
    inet 172.16.10.5/24 brd 172.16.10.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:feeb:2dd0/64 scope link
        valid_lft forever preferred_lft forever
```

## 3. Kali client

Vérification carte réseau:



Clavier français:

```
File Actions Edit View H
(kali@kali)-[~]
└─$ setxkbmap fr
```

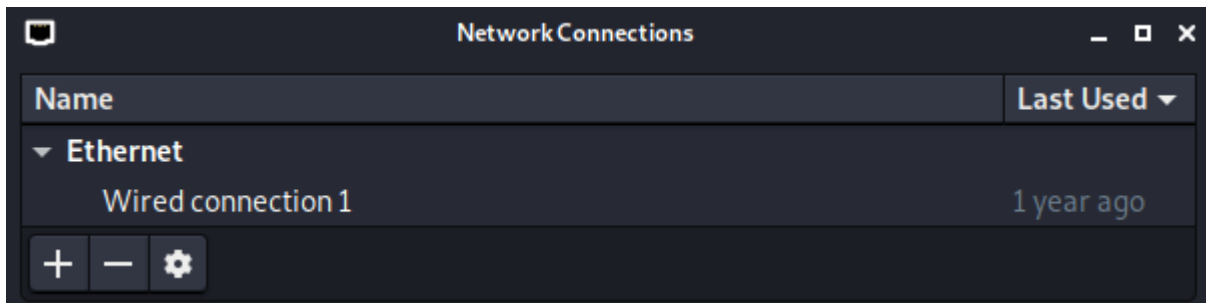
Je n'ai pas la bonne adresse ip dans la machine kali, il faut donc ajouter une adresse ip et la configurer en manuel pour que la kali soit dans le réseau lan-in:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast s
group default qlen 1000
    link/ether 08:00:27:cf:9b:43 brd ff:ff:ff:ff:ff:ff
```

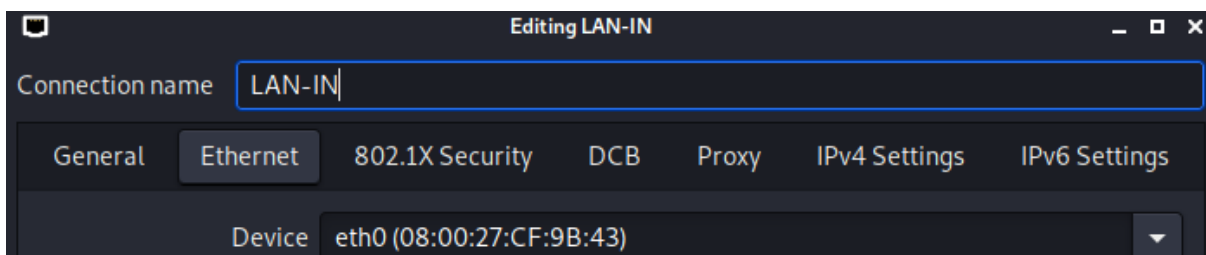
Pour ajouter un adresse ip en manuel kali:

Aller dans le menu Applications / Settings / NetworkConnections

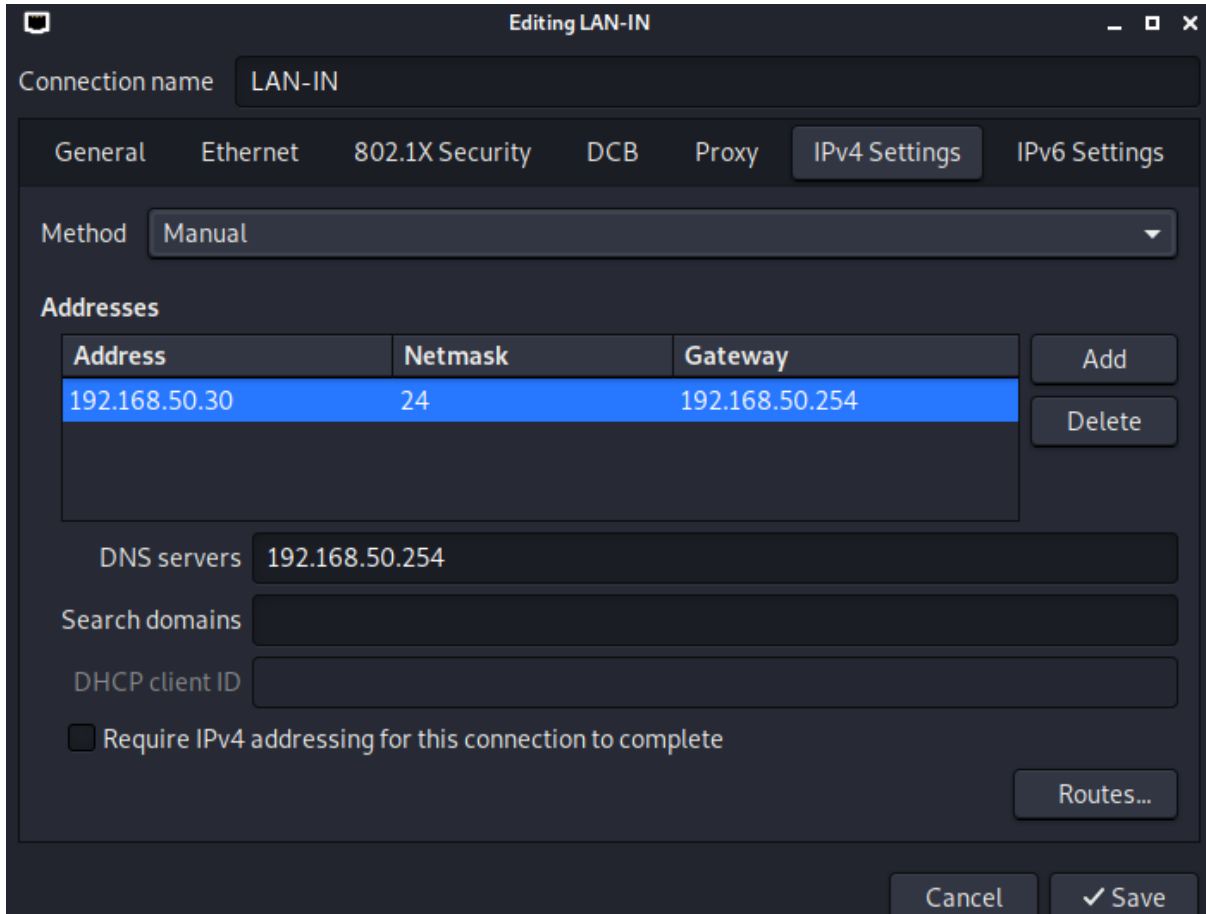
Puis cliquer sur le “+” pour éditer une nouvelle connexion.



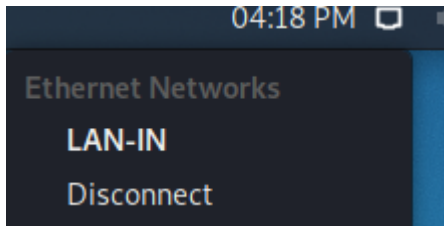
Ethernet: Choisir son nom de connexion et choisir la bonne carte réseau.



IPV4 Settings: Mettre en manuel puis ajouter la bonne adresse ip et la bonne passerelle ainsi que le DNS.



On peut donc apercevoir le réseau LAN-IN que je viens de créer.



Puis relancer la connexion:

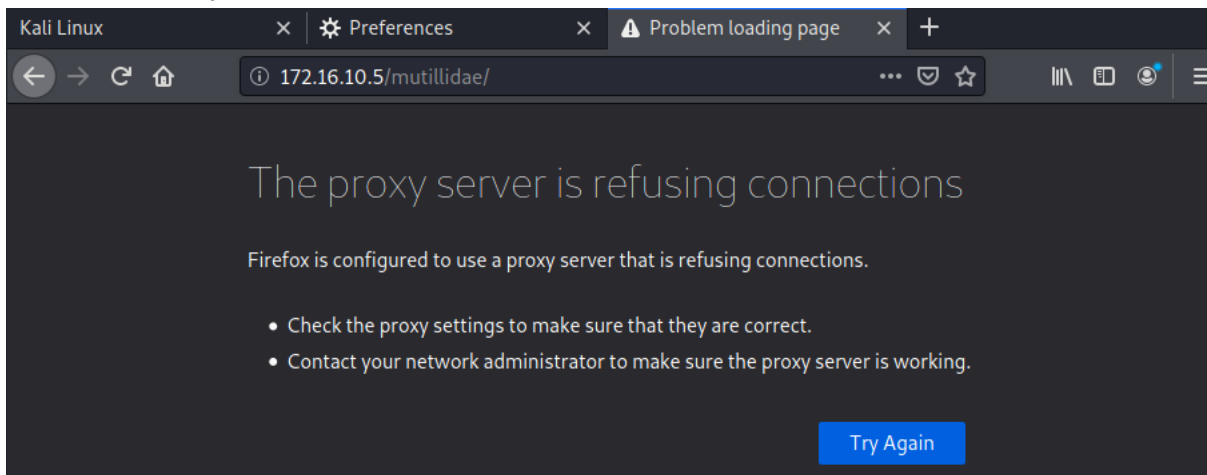
```
(kali㉿kali)-[~]
└─$ sudo /etc/init.d/networking restart
Restarting networking (via systemctl): networking.service.
```

Et je vérifie l'adresse IP de ma carte réseau:

```
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP
group default qlen 1000
    link/ether 08:00:27:cf:9b:43 brd ff:ff:ff:ff:ff:ff
    inet 192.168.50.30/24 brd 192.168.50.255 scope global noprefixroute eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::ca3b:fdb1:c981:4a29/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

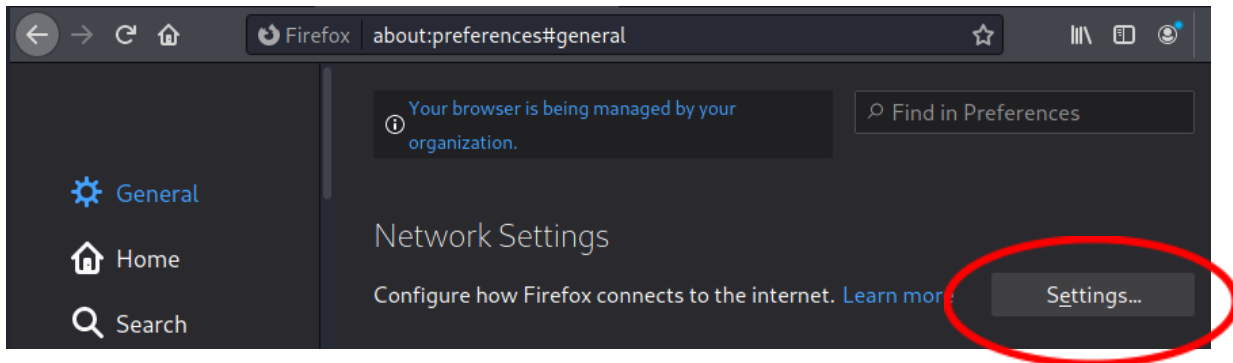
Désormais toutes mes machines sont correctement configurées.

Ensuite la logique serait de se connecter directement sur le serveur mutillidae et de commencer le TP. Mais lorsque que l'on essaye, le navigateur nous affiche un message d'erreur du proxy qui nous dit qu'il a refusé la connexion au serveur.

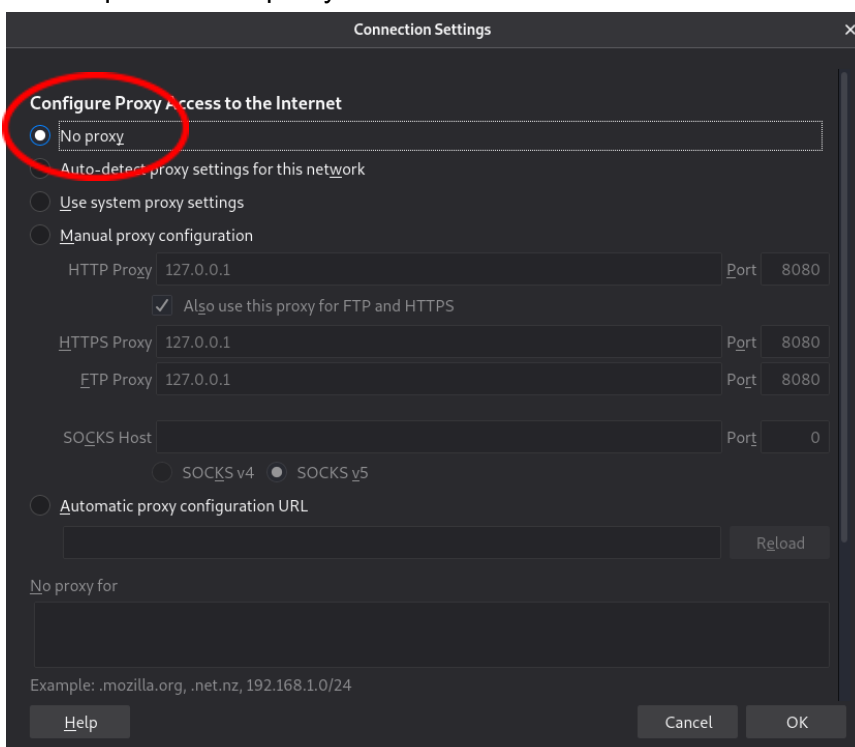


Après cela on doit désactiver le proxy sur notre kali car :

Pour cela il suffit d'ouvrir son navigateur web pour ma part Mozilla, aller dans les paramètres "Général" puis dans "Network Settings" et cliquer sur "Settings...".



Puis cliquer sur "No proxy" afin de le désactiver:



Réessayer de se connecter au serveur mutillidae depuis son navigateur web avec le proxy désactivé:



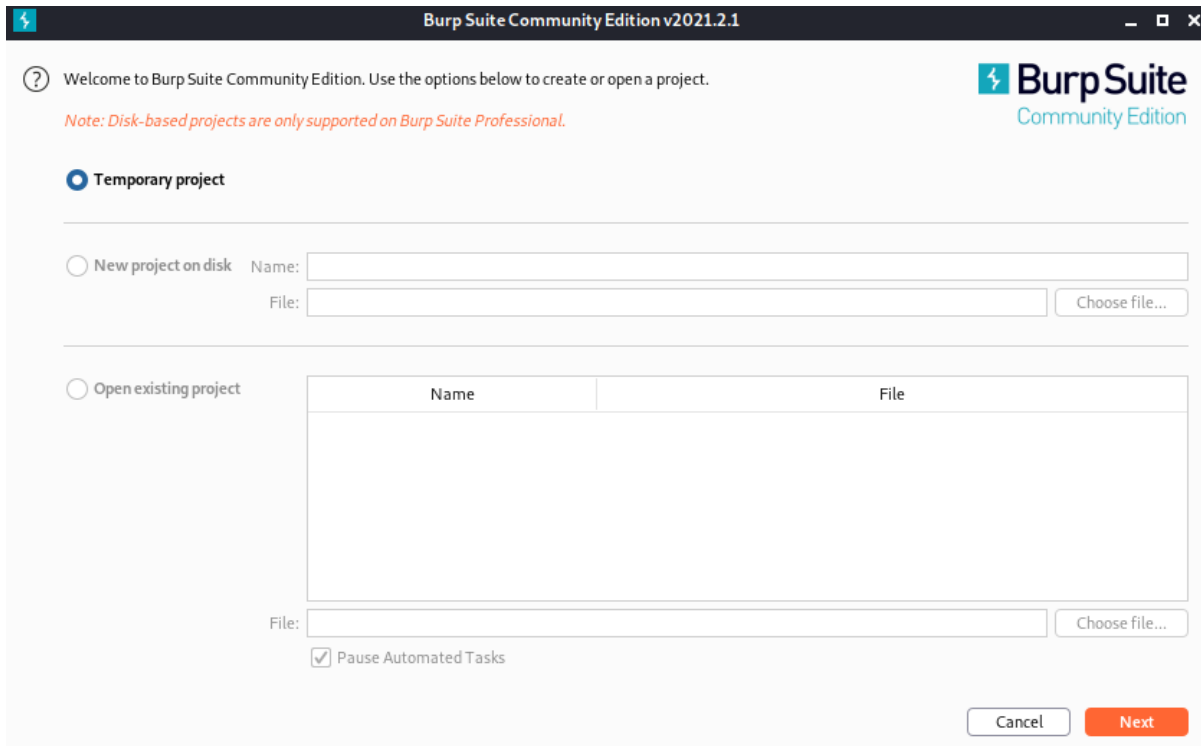
Cela fonctionne correctement, notre contexte est alors bien configuré.

## TP02

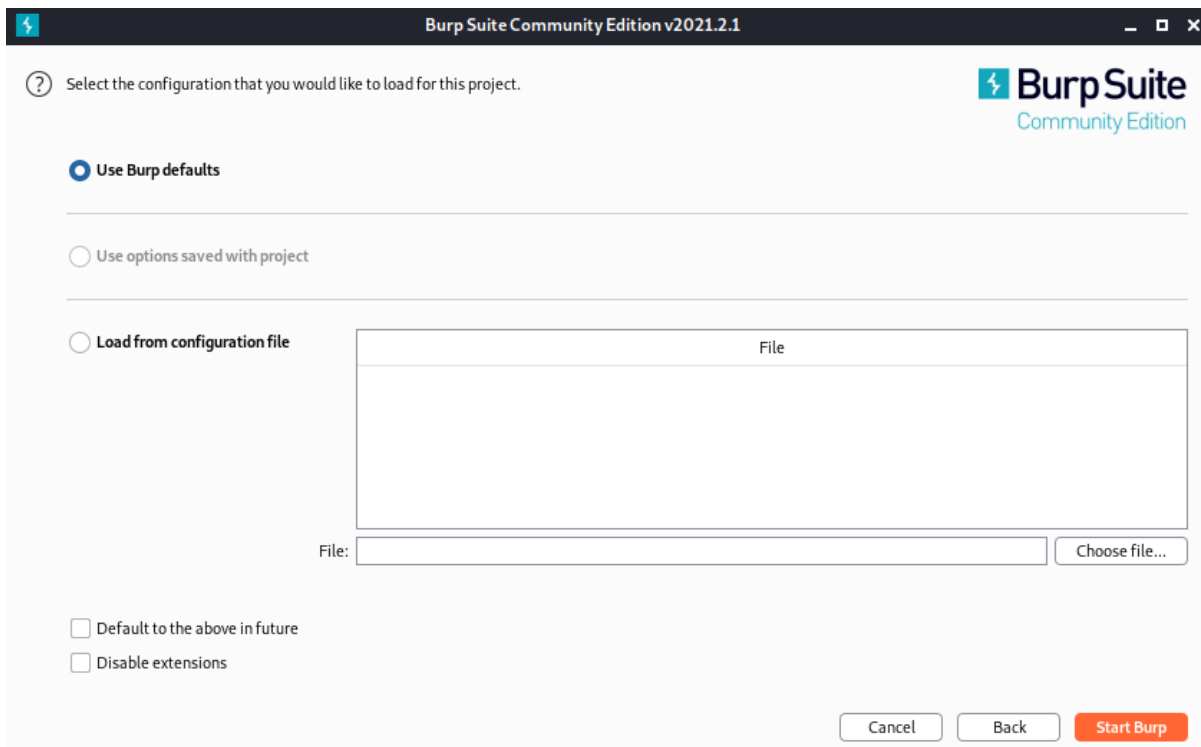
### APPRENTISSAGE 1 : énumération des logins

#### Installation de l'extension Wsdler

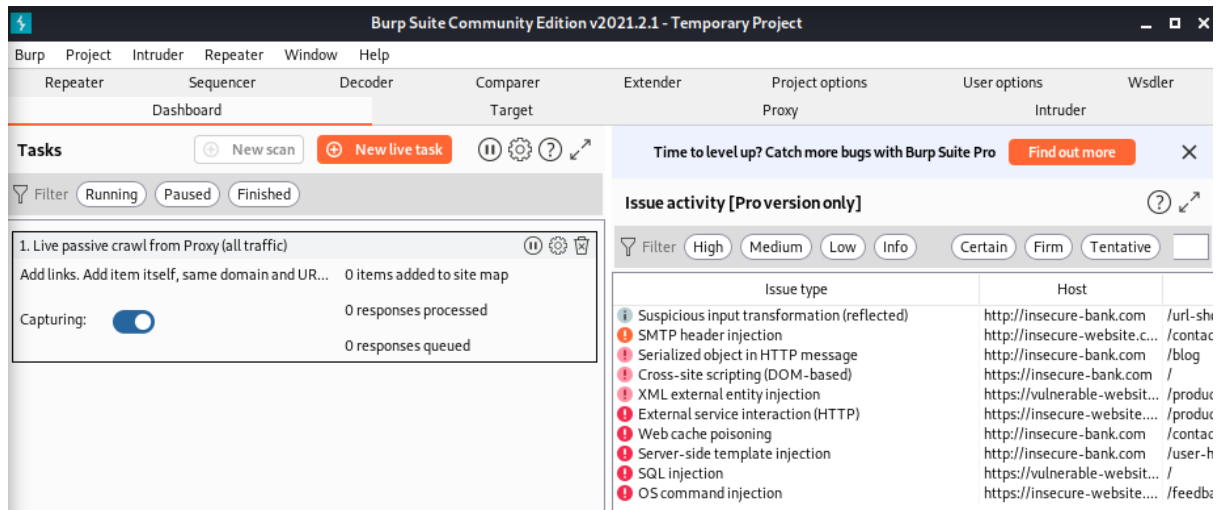
Lancer BurpSuite puis configurer les paramètre "Temporary project" :



Puis Use "Burp defaults" et "StartBurp":



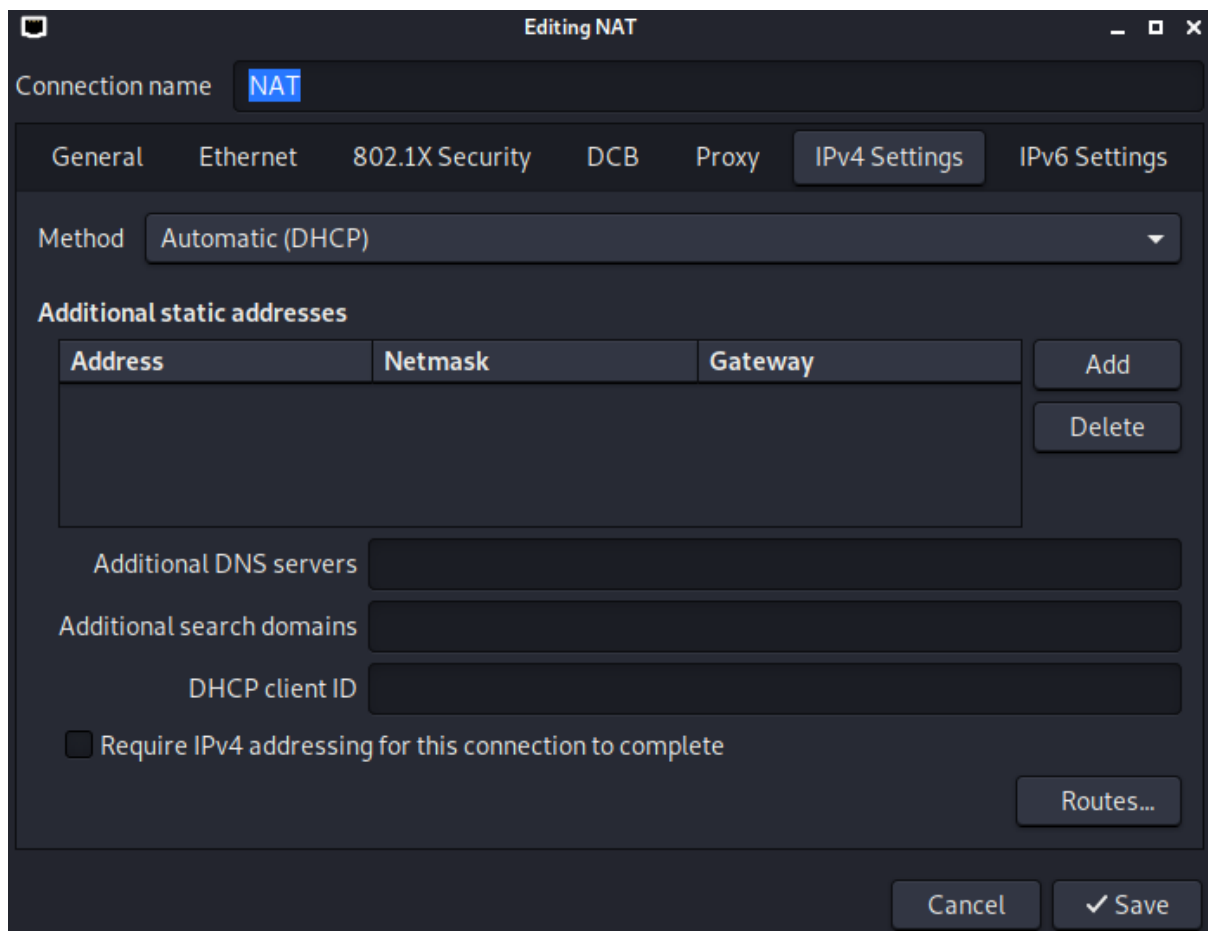
Et donc voici la page d'accueil de BurpSuite:



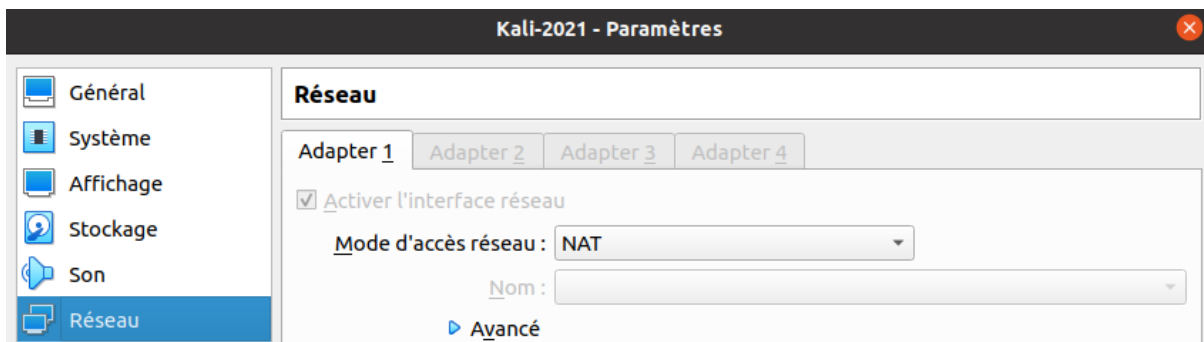
Pour installer l'extension "Wsdler" il faut alors changer son réseau pour avoir Internet.

Puis créer un connexion NAT sur notre VM Kali afin de se connecter directement au réseau du lycée et avoir accès à Internet, nous allons donc sortir de notre contexte.

Pour créer notre connexion NAT aller dans les mêmes paramètres que lorsque l'on a créé le réseau LAN-IN. Mais il suffit de modifier IPV4 en Automatic. On va donc directement demander une adresse au serveur DHCP de notre réseau.



Modifier aussi les paramètres de la VM en NAT

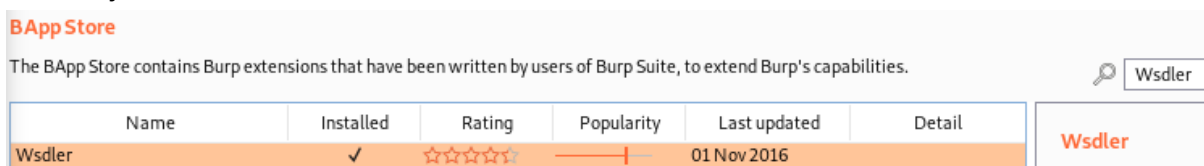


Puis sur Burp Suite aller dans "Extender", puis dans "BApp Store":

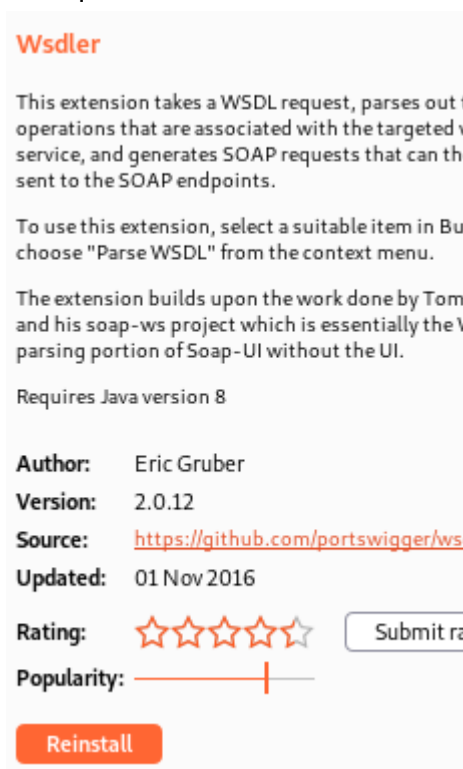


Puis rechercher l'extension "Wsdler":

Elle est déjà installée mais par précaution je la réinstalle afin de la mettre à jour s' il y a des mises à jour.

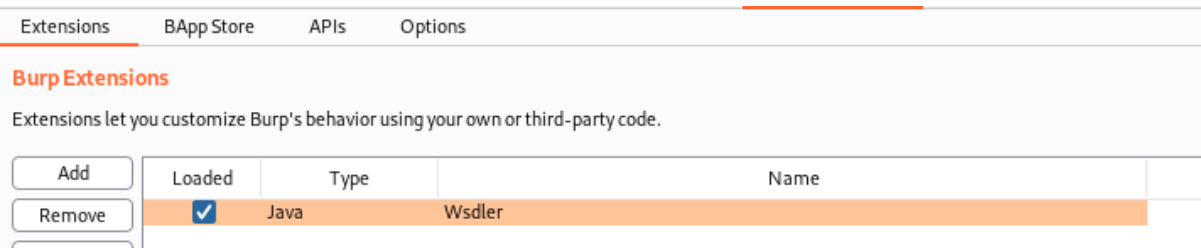


Je clique donc sur "reinstall":





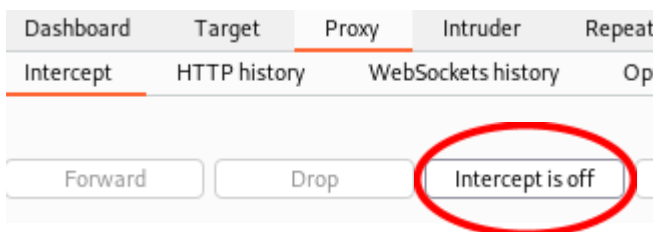
On vérifie qu'elle soit bien installée:



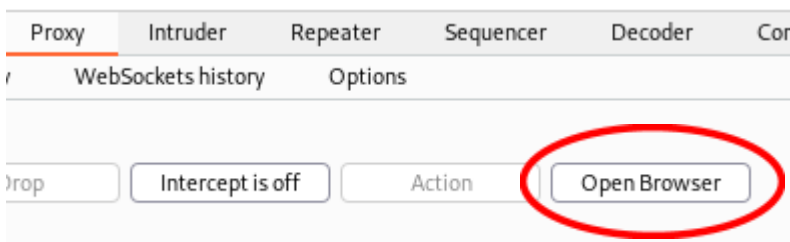
Puis remodifier la connexion en LAN-IN.

### Test d'une requête et d'une réponse sur un login inexistant

Toujours sur BurpSuite aller dans l'onglet proxy. Puis cliquer sur "Intercept is On" pour le mettre en Off.



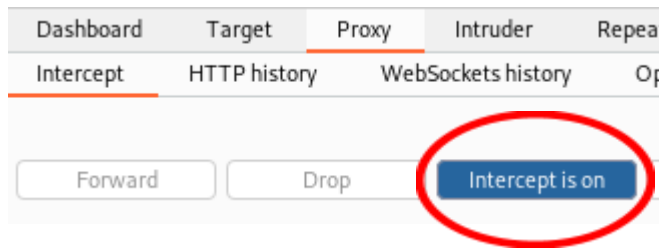
Puis accéder à l'application Mutillidae en cliquant sur **Open Browser**.



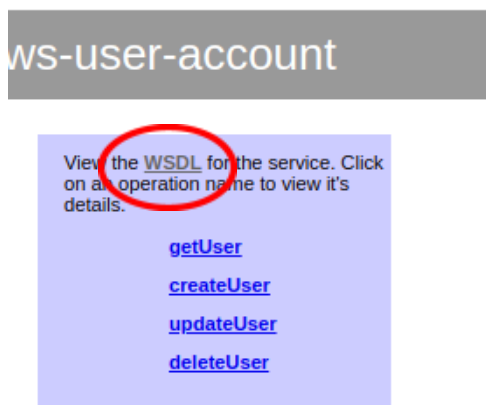
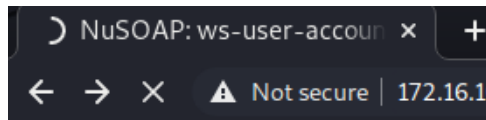
Puis aller dans OWASP 2017 > A2 > Username Enumeration > Lookup User



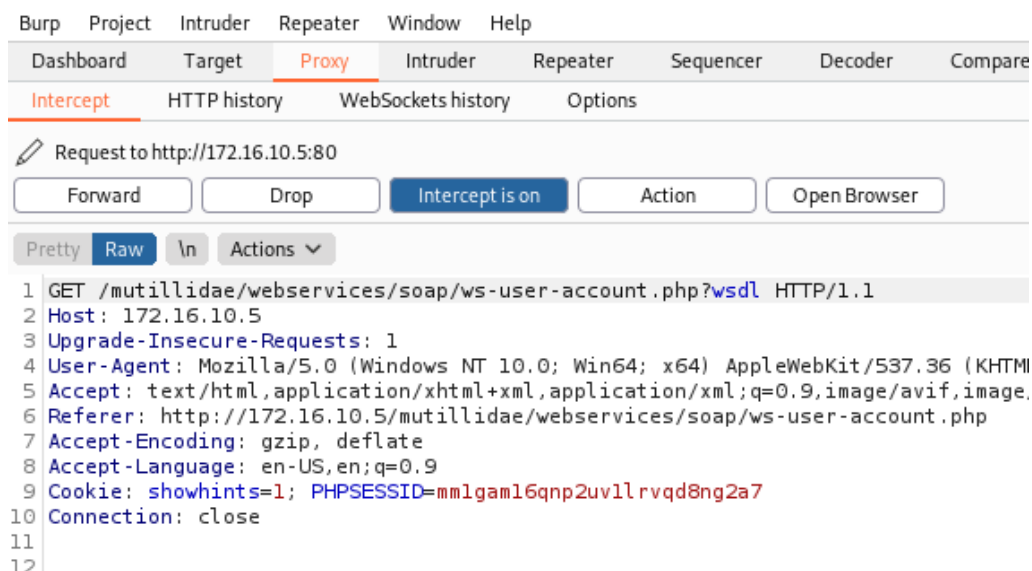
Ensuite remettre le proxy de BurpSuite en intercept on



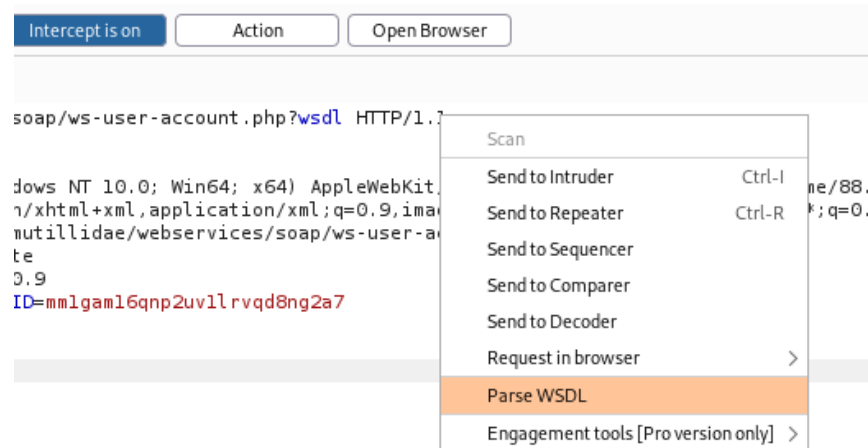
Cliquer sur WSDL



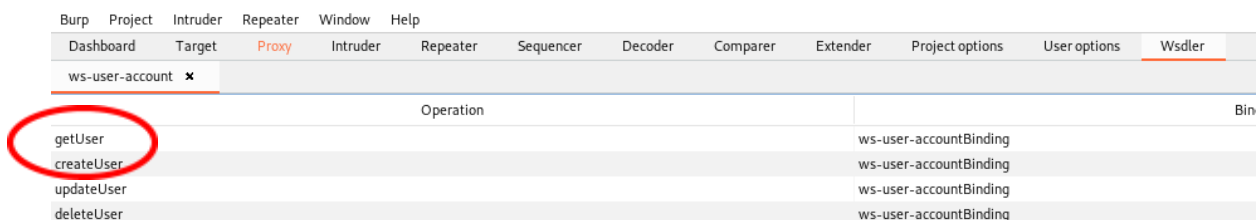
On est directement renvoyé sur le Burp dans l'onglet Raw.



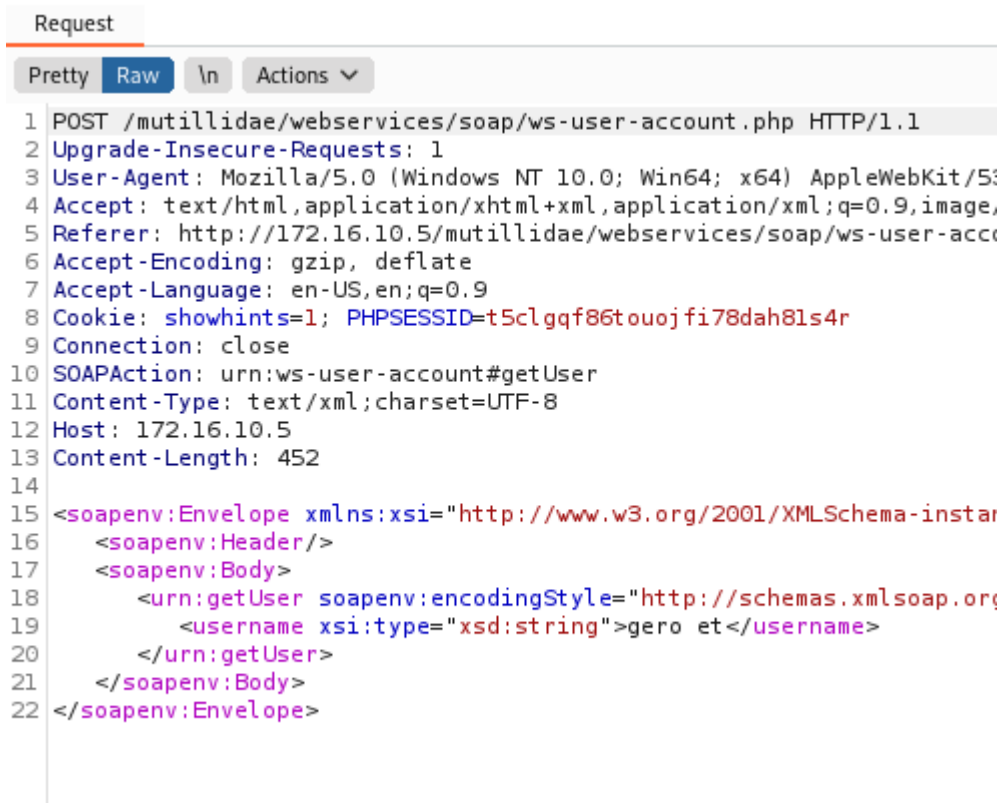
Puis faire un clique droit dans l'onglet "Raw" et cliquer sur Parse WSDL:



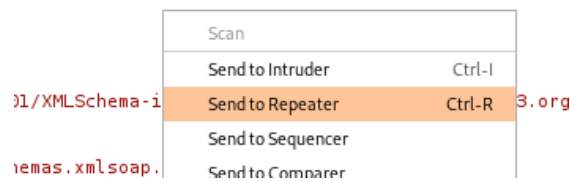
Aller dans l'onglet Wsdler, je vérifie que j'ai bien getUser:



Puis on peut voir la requête de getUser. Dans la balise username on peut apercevoir "gero et". En effet ce compte est en réalité un login qui n'existe pas dans la liste des logins valides des comptes déjà existants.



Puis faire un clique droit ou “ctrl+R” sur la fenêtre et aller dans “Send to Repeater”:



Cette étape permet d’envoyer la requête dans l’onglet Repeater, cet onglet nous permet de modifier ou forger notre requête puis de la tester autant de fois que l’on souhaite avant de l’envoyer au serveur mutillidae.

Voici donc la requête:

```

1 POST /mutillidae/webservices/soap/ws-user-account.php HTTP/1.1
2 Upgrade-Insecure-Requests: 1
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/88.0.4324.150 Safari/537.36
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
5 Referer: http://172.16.10.5/mutillidae/webservices/soap/ws-user-account.php
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: showhints=1; PHPSESSID=mm1gam16qnp2uv1lrvqd8ng2a7
9 Connection: close
10 SOAPAction: urn:ws-user-account#getUser
11 Content-Type: text/xml; charset=UTF-8
12 Host: 172.16.10.5
13 Content-Length: 452
14
15 <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
16   xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:urn="urn:ws-user-account">
17   <soapenv:Header/>
18   <soapenv:Body>
19     <urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
20       <username xsi:type="xsd:string">gero et</username>
21     </urn:getUser>
22   </soapenv:Body>
23 </soapenv:Envelope>

```

Dans cet onglet il y a 2 parties:

- **Request** qui permet de voir et modifier notre requête
- **Response** qui permet de voir le flux traité après avoir envoyé la requête

On va donc envoyer la requête au serveur en cliquant sur “send”:

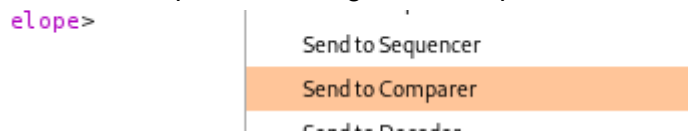
```

1 HTTP/1.1 200 OK
2 Date: Sun, 27 Nov 2022 20:52:37 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 X-SOAP-Server: NuSOAP/0.9.5 (1.123)
8 Vary: Accept-Encoding
9 Content-Length: 559
10 Connection: close
11 Content-Type: text/xml; charset=ISO-8859-1
12
13 <?xml version="1.0" encoding="ISO-8859-1"?>
14   <SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
15     <SOAP-ENV:Body>
16       <ns1:getUserResponse xmlns:ns1="urn:ws-user-account">
17         <return xsi:type="xsd:string">
18           <accounts message="User gero et does not exist" />
19         </return>
20       </ns1:getUserResponse>
21     </SOAP-ENV:Body>
22   </SOAP-ENV:Envelope>

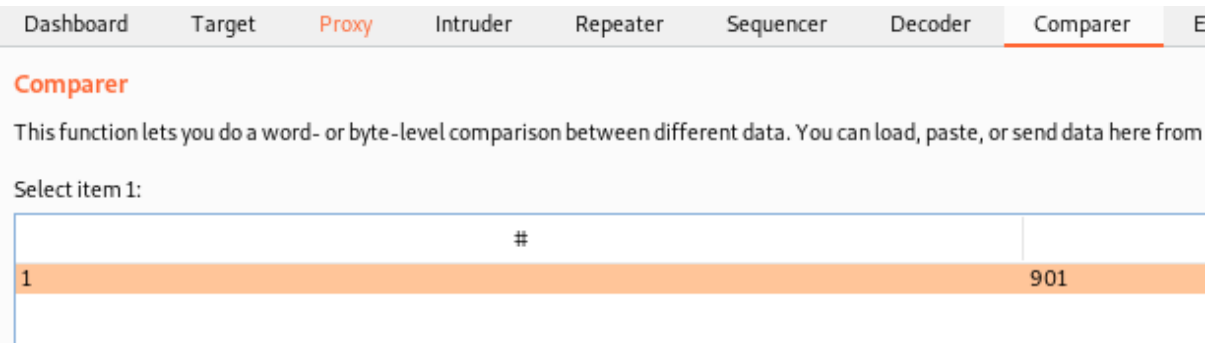
```

Le serveur nous dit qu'il n'existe pas d'utilisateur.

On peut aussi stocker la requête afin de la comparer à une autre en faisant un clic droit > Send to Comparer sur l'onglet de la réponse.



Pour retrouver notre requête il suffit d'aller dans l'onglet "Comparer", notre première requête y est stockée.



#### A1.Q1 :

Commande à exécuter depuis l'onglet Proxy jusqu'à l'onglet Comparer.

Aller dans

- **BurpsSuite > Proxy/Intercept off >**
  - **Open Browser > 172.16.10.5/mutillidae > OWASP 2017/A2/Username Enumeration/Lookup User >**
- **BurpsSuite > Proxy/Intercept on >**
  - **Revenir sur le lien Lookup User et cliquer sur "WSDL" >**
- **BurpsSuite > Proxy/Intercept/clic droit/Parse WSDL > Wsdler/GetUser > clic droit/Send to Repeater > Repeater >**
  - **on peut modifier la requête si on le souhaite dans /Request**
  - **pour stocker la requête /clic droit /Send to Comparer**
  - **pour l'envoyer /Send on a la réponse dans /Response**
- **BurpsSuite > Comparer**
  - **pour comparer 2 requêtes faire /Words**

#### Test d'une requête et d'une réponse sur un login existant

Nous devons tout d'abord créer 2 utilisateurs avec des mdp sur mutillidae.

Pour créer un utilisateur mutillidae:

A screenshot of a login form. At the top is a pink button labeled 'Please sign-in'. Below it are two input fields: 'Username' and 'Password'. A blue 'Login' button is positioned below the password field. At the bottom, there is a link that says 'Dont have an account? Please register here', which is circled in red.

Je créer donc 2 comptes :

1er

- username : utilisateur1
- Password : utilisateur1

2eme

- username : utilisateur2
- Password : utilisateur2

**Please choose your username, password and signature**

<b>Username</b>	<input type="text" value="utilisateur1"/>	
<b>Password</b>	<input type="password" value="••••••••••"/>	<a href="#">Password Generator</a>
<b>Confirm Password</b>	<input type="password" value="••••••••••"/>	
<b>Signature</b>	<input type="text" value="Created by Matthias Hautin"/>	

Je fais donc la même chose pour le deuxième compte pour le créer.

On va donc tester comme dans l'étape précédente mais cette fois-ci avec l'un des logins que l'on vient de créer par exemple utilisateur1.

Je refais donc toute la manipulation.

Lorsque on arrive dans l'étape de Repeater on doit donc modifier la requête et mettre le user "utilisateur1" que l'on vient de créer à la place de "gero et".

Voici la modification:

```
Request
Pretty Raw \n Actions v
1 POST /mutillidae/webservices/soap/ws-user-account.php HTTP/1.1
2 Upgrade-Insecure-Requests: 1
3 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,
5 Referer: http://172.16.10.5/mutillidae/webservices/soap/ws-user-account.php
6 Accept-Encoding: gzip, deflate
7 Accept-Language: en-US,en;q=0.9
8 Cookie: showhints=1; PHPSESSID=t5clgqf86touojfi78dah8ls4r
9 Connection: close
10 SOAPAction: urn:ws-user-account#getUser
11 Content-Type: text/xml;charset=UTF-8
12 Host: 172.16.10.5
13 Content-Length: 457
14
15 <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="h
16 <soapenv:Header/>
17 <soapenv:Body>
18 <urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
19 <username xsi:type="xsd:string">
20 utilisateur1
21 </username>
22 </urn:getUser>
23 </soapenv:Body>
24 </soapenv:Envelope>
```

Voici la Réponse:

### Response

```
1 HTTP/1.1 200 OK
2 Date: Sun, 27 Nov 2022 23:12:08 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 X-SOAP-Server: NuSOAP/0.9.5 (1.123)
8 Vary: Accept-Encoding
9 Content-Length: 666
L0 Connection: close
L1 Content-Type: text/xml; charset=ISO-8859-1
L2
L3 <?xml version="1.0" encoding="ISO-8859-1"?>
  <SOAP-ENV:Envelope SOAP-ENV:encodingStyle="http://schemas.x
    <SOAP-ENV:Body>
      <ns1:getUserResponse xmlns:ns1="urn:ws-user-account">
        <return xsi:type="xsd:xml">
          <accounts message="Results for utilisateur1">
            <account>
              <username>
                utilisateur1
              </username>
              <signature>
                Created by Matthias Hautin
              </signature>
            </account>
          </accounts>
        </return>
      </ns1:getUserResponse>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Burp trouve donc bien l'utilisateur1 et on a même la signature.

Envoyer la requête dans le Comparer: clique droit/ Send to Comparer

Aller dans "Comparer" et cliquer sur "Words":

Ceci va donc comparer notre ancienne requête avec "gero" et notre nouvelle requête avec "utilisateur1" et on va donc voir les différences comme par exemple l'heure d'envoi de notre requête au serveur ou encore la réponse:

The screenshot shows the 'Word compare' tool in Burp Suite, comparing two HTTP responses. The tool is titled 'Word compare of #1 and #3 (11 differences)'. The left pane shows the first response (Length: 901) and the right pane shows the second response (Length: 1,008). The differences are highlighted in red and green. The key differences are:

- Date: Sun, 27 Nov 2022 20:52:37 GMT (Response 1) vs Date: Sun, 27 Nov 2022 23:12:08 GMT (Response 2)
- Content-Length: 559 (Response 1) vs Content-Length: 666 (Response 2)
- Signature: Created by Matthias Hautin (Response 1) vs Created by Matthias Hautin (Response 2)

The tool also shows a key: Modified (red), Deleted (blue), Added (green).

Réponse première requête gero:

Word compare of #1 and

Length: 901  Text  Hex

```
<return xsi:type="xsd:string"><accounts message="User gero et does not exist" /></return></ns1:getUserResponse></SOAP-ENV:Body></SOAP-ENV:Envelope>
```

Key: Modified Deleted Added

Réponse deuxième requête utilisateur1:

3 (11 differences)

Length: 1,008  Text  Hex

```
<return xsi:type="xsd:string"><accounts message="Results for utilisateur1"><account><username>utilisateur1</username><signature>Created by Matthias Hautin</signature></account></accounts></return>
```

Supprimer

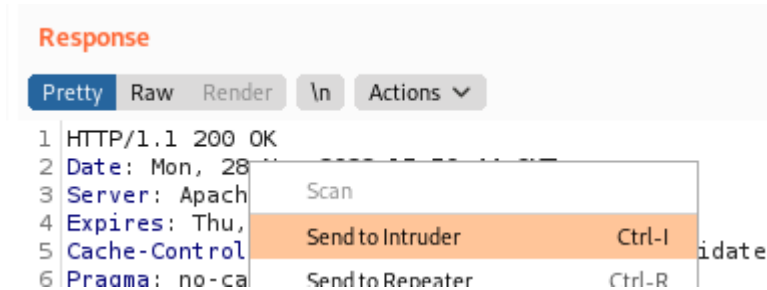
On peut donc remarquer les différences.



## Énumération des logins

Aller dans l'onglet Repeater de BurpSuite

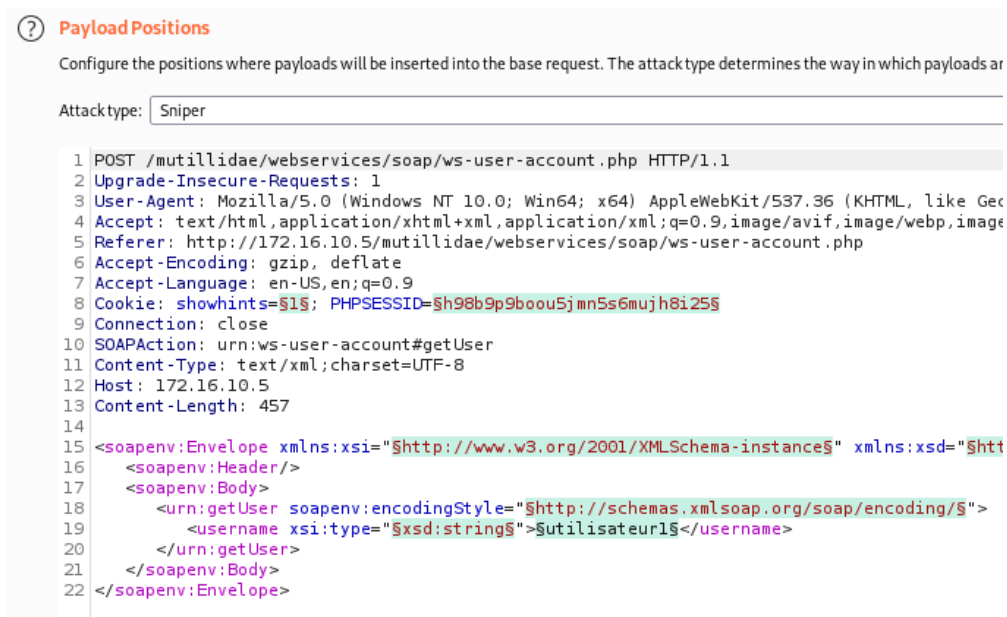
Faire un clic droit sur l'onglet de la réponse correct avec utilisateur1 puis cliquer sur "Send to the Intruder" ou faire "Ctrl+I":



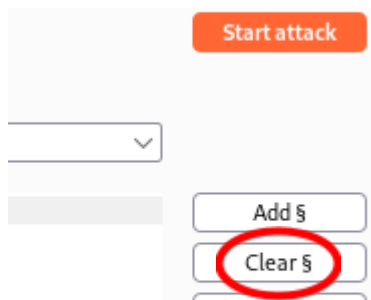
Se rendre dans l'onglet intruder/Positions puis cliquer sur "Clear §":



Voilà notre requête que l'on vient d'envoyer.



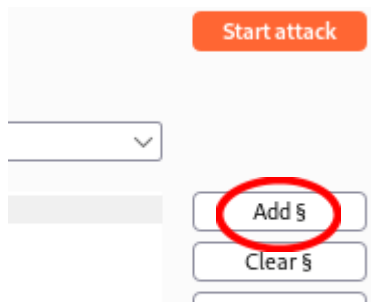
Cliquer sur "Clear §":



Puis après avoir clear faire un double clique sur le login "utilisateur1", on peut voir apparaître la valeur en surligné jaune:

```
15 <soapenv:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-insta
16 <soapenv:Header/>
17 <soapenv:Body>
18 <urn:getUser soapenv:encodingStyle="http://schemas.xmlsoap.or
19 <username xsi:type="xsd:string">utilisateur1</username>
20 </urn:getUser>
21 </soapenv:Body>
22 </soapenv:Envelope>
```

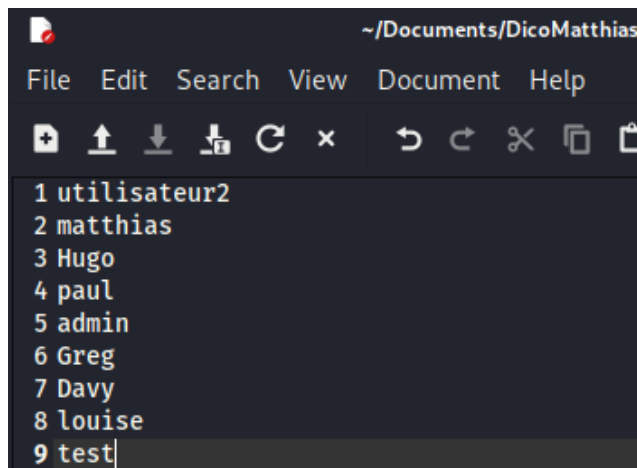
Puis cliquer sur "Add §"



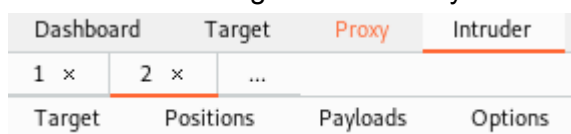
On peut remarquer que utilisateur1 est en surligné vert et il est entouré de 2 "§":

```
.e="http://schemas.xmlsoap.org
j">§utilisateur1§</username>
```

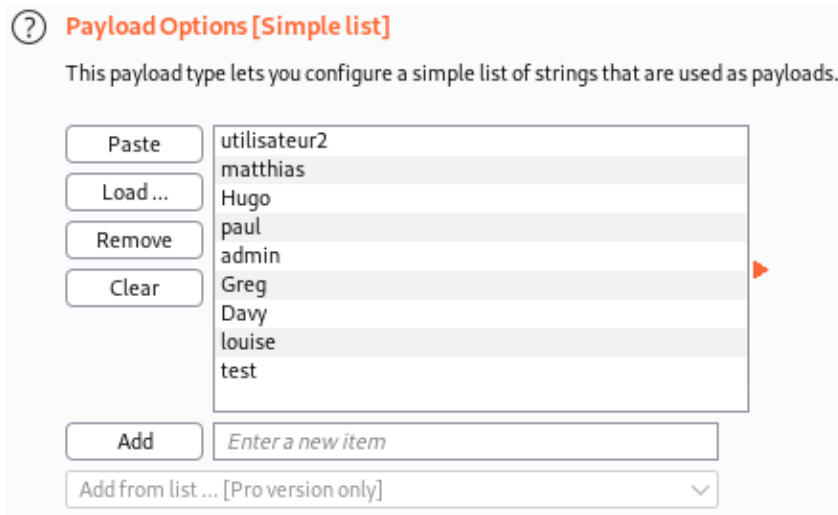
Ensuite ouvrir un éditeur de texte par exemple Text editor, je l'ai appelé DicoMatthias. Y ajouter plusieurs login et le save dans Kali / Documents:



Puis aller dans l'onglet intruder/Payload:



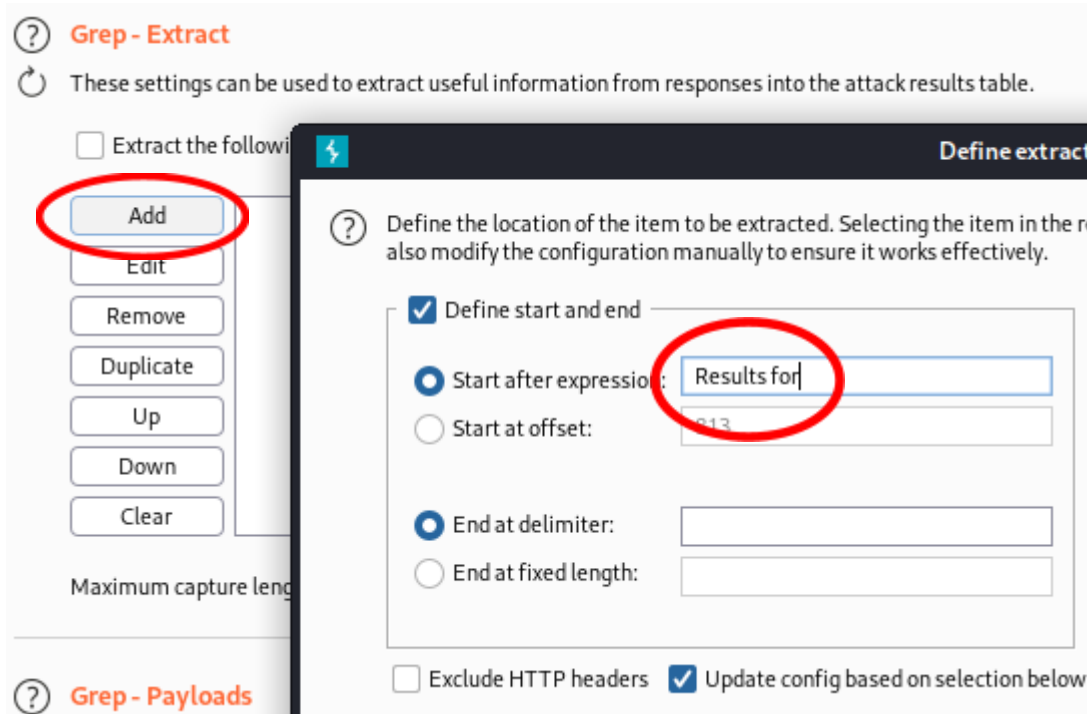
Puis Payload Options [Simple list] et y insérer notre liste:



Après aller dans l'onglet intruder/Option:



Aller dans la rubrique Grep Extract, cliquer sur Add, puis saisir "Results for" ce qui correspond à une chaîne de caractère d'un login correct.



Puis cliquer sur "Start Attack":



Voici donc la liste d'énumération des logins:

Request ^	Payload	Status	Error	Timeout	Length	Results for	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	1008	utilisateur1"><account>...	
1	utilisateur2	200	<input type="checkbox"/>	<input type="checkbox"/>	1007	utilisateur2"><account>...	
2	matthias	200	<input type="checkbox"/>	<input type="checkbox"/>	902		
3	Hugo	200	<input type="checkbox"/>	<input type="checkbox"/>	898		
4	paul	200	<input type="checkbox"/>	<input type="checkbox"/>	898		
5	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	976	admin"><account><user...	
6	Greg	200	<input type="checkbox"/>	<input type="checkbox"/>	898		
7	Davy	200	<input type="checkbox"/>	<input type="checkbox"/>	898		
8	louise	200	<input type="checkbox"/>	<input type="checkbox"/>	900		
9	test	200	<input type="checkbox"/>	<input type="checkbox"/>	898		

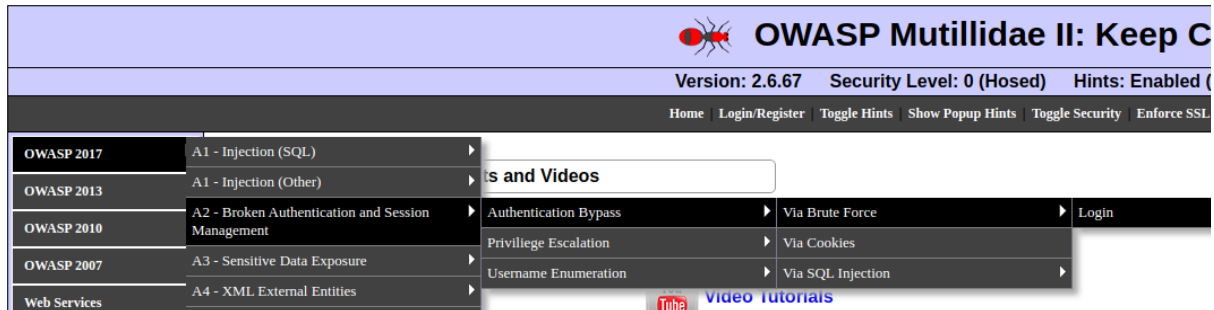
## Force brute d'un mot de passe

### 1) Préparation de l'attaque

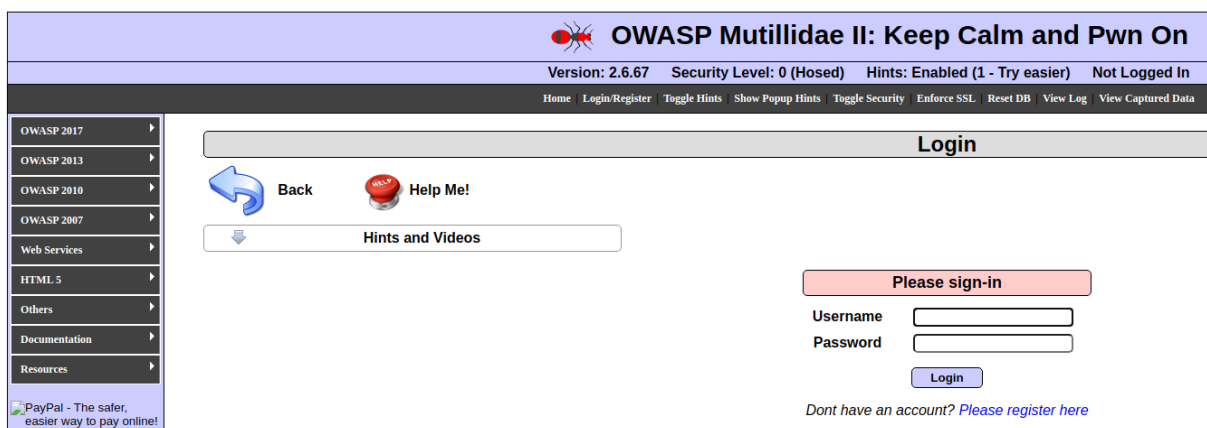
Une attaque de force Brut consiste à trouver un mot de passe d'un utilisateur existant à l'aide d'un dictionnaire de mots de passe que l'on teste en boucle. Nous allons créer nous même un dictionnaire, ne possédant pas un dictionnaire important de mots de passe.

Aller sur Burp Suite mettre Intercept en mode "Intercept off", aller dans mutillidae avec Open Browser, mettre le level Security : 0

Puis aller dans OWASP 2017 > A2 > Authentication Bypass > Via Brute Force > Login:



Page Login pour Brute Force:



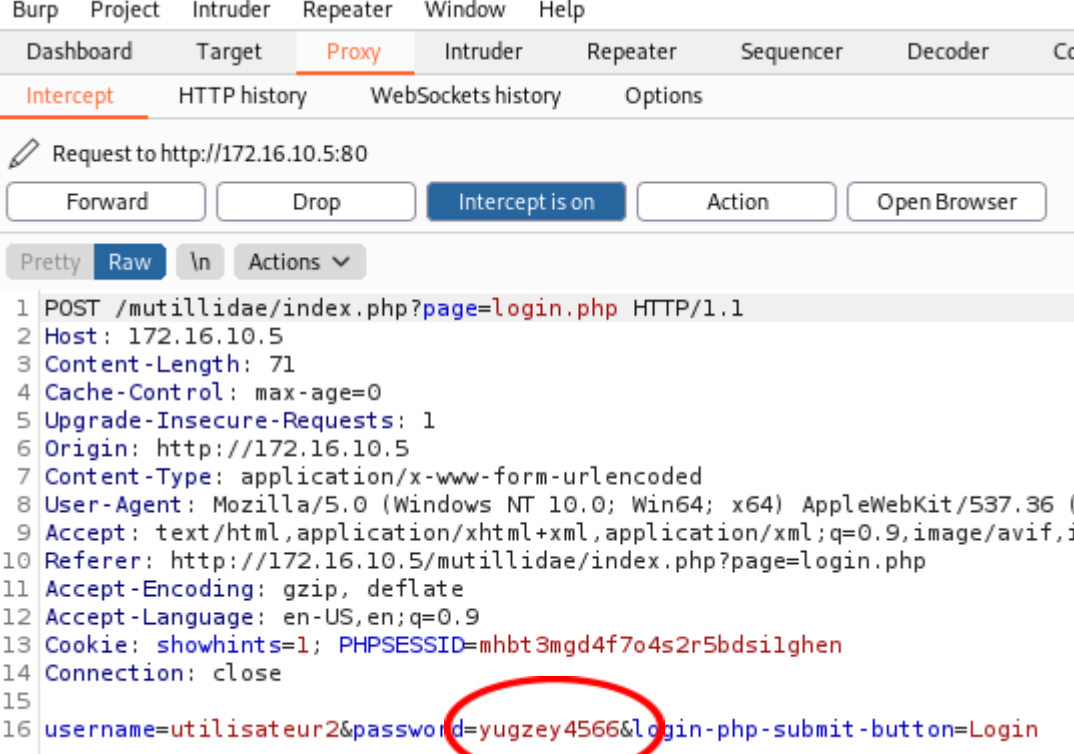
Retourner sur Burp Suite et mettre "Intercept On".

Cette partie consiste à créer une variable, sur mutillidae saisir:

Username : utilisateur2

Password : yugzey4566 (n'importe quel mot de passe erroné)

Sur Burp Suite on peut donc apercevoir la requête:



Burp Project Intruder Repeater Window Help

Dashboard Target Proxy Intruder Repeater Sequencer Decoder Co

Intercept HTTP history WebSockets history Options

Request to http://172.16.10.5:80

Forward Drop Intercept is on Action Open Browser

Pretty Raw \n Actions

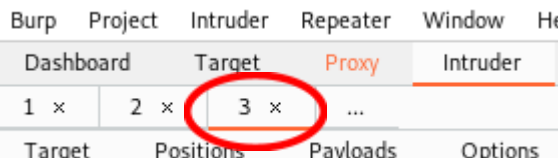
```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: 172.16.10.5
3 Content-Length: 71
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.16.10.5
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (
9 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,i
10 Referer: http://172.16.10.5/mutillidae/index.php?page=login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: showhints=1; PHPSESSID=mhbt3mgd4f7o4s2r5bdsilghen
14 Connection: close
15
16 username=utilisateur2&password=yugzey4566&login-php-submit-button=Login
```

On retrouve le mot de passe que j'ai utilisé

## 2) Lancement de l'attaque

Faire un clique droit vers "Send to Intruder" comme déjà réalisé.

Puis aller dans Intruder et y remarque un nouvel onglet "3 x":



Burp Project Intruder Repeater Window He

Dashboard Target Proxy Intruder

1 x 2 x 3 x ...

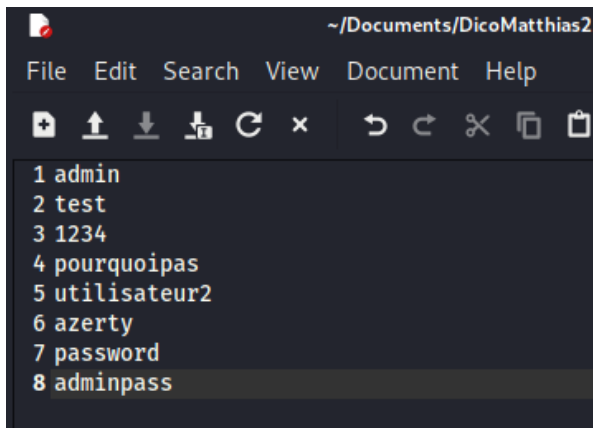
Target Positions Payloads Options

Puis l'onglet position et "Clear §" comme précédemment réalisé.

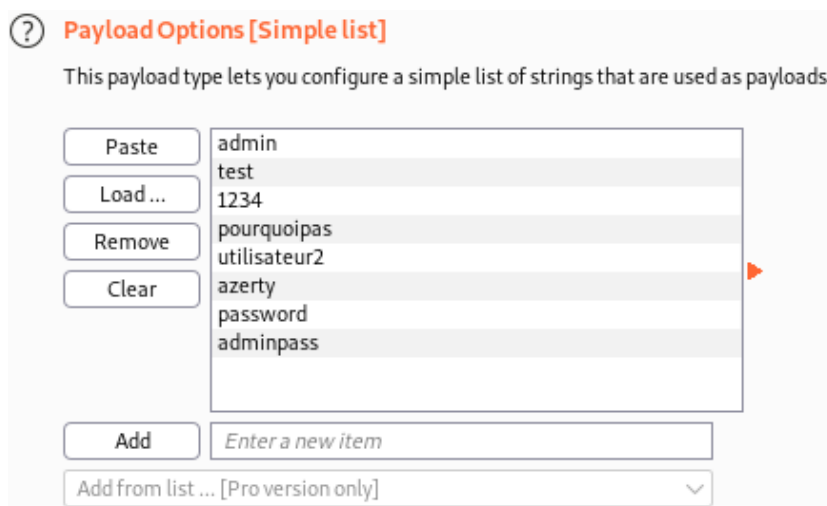
Puis double cliquer sur le password "yugzey4566" et faire "Add §" pour créer notre variable.

```
username=utilisateur2&password=§yugzey4566§&l
```

Puis créer un nouveau dictionnaire avec différents mots de passe ainsi que le mot de passe correct de utilisateur2 = "utilisateur2":



Puis charger les mots de passes dans "Payload/Payload Options" comme précédemment effectué:



Et on lance l'attaque:

Attack Save Columns

Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request ^	Payload	Status	Error	Timeout	Length	Comment
0		200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
1	admin	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
2	test	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
3	1234	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
4	pourquoiipas	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
5	utilisateur2	302	<input type="checkbox"/>	<input type="checkbox"/>	382	
6	azerty	200	<input type="checkbox"/>	<input type="checkbox"/>	55357	
7	password	200	<input type="checkbox"/>	<input type="checkbox"/>	55357	
8	adminpass	200	<input type="checkbox"/>	<input type="checkbox"/>	55357	

On a une réponse 302 ce qui signifie que le mot de passe "utilisateur2" correspond à la bonne authentification.

Bien évidemment qu' en conditions d'audit de sécurité nous utiliserons un dictionnaire beaucoup plus volumineux avec plus de mots de passe.

**CHALLENGE 1 : énumération des logins en mode sécurisé**

**D1.Q1.** Fermer puis relancer BurpSuite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant les étapes "Test d'une requête et d'une réponse sur un login existant", "Énumération des logins". Montrez votre travail par des copies d'écran (avec votre nom) et des explications.

Tout d'abord créer un nouveau user dans Mozilla que je vais appeler:

username : Matthias

Password : Matthias1234

**Please choose your username, password and signature**

**Username**

Matthias

**Password**

●●●●●●●●●●

[Password Generator](#)

**Confirm Password**

●●●●●●●●●●

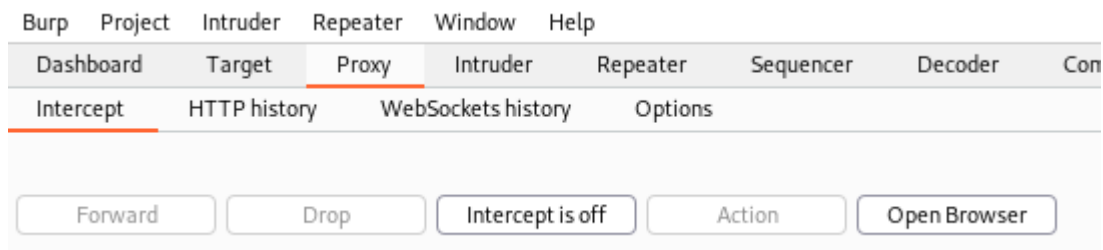
**Signature**

Create by Matthias Hautin

Create Account

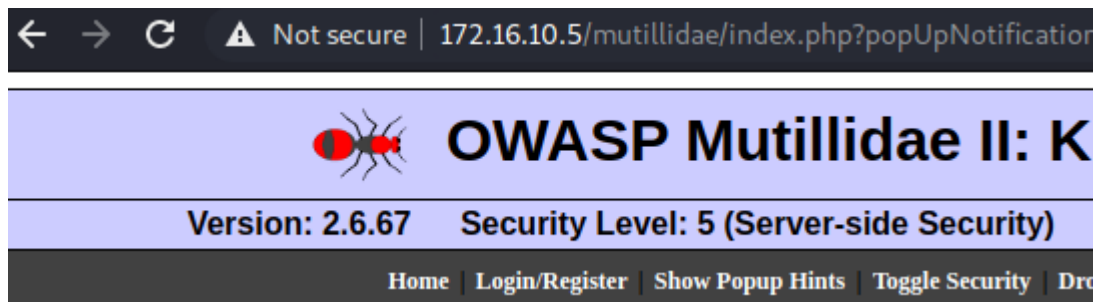
Puis aller sur Burp Suite et aller dans Prox/Intercept, mettre "Intercept is off".

Puis cliquer sur Open Browser:



Puis insérer le lien sur serveur mutillidae : "172.16.10.5/mutillidae"

Mettre la sécurité au level 5 cliquer sur Toggle Security:



Ensuite aller dans OWASP 2017 > A2 > Authentication Bypass > Via Brute Force > Login:

Je remet "Intercept is on"

Comme prévu on arrive sur la page de login.

Dans cette page je dois m'identifier sans insérer un mauvais mot de passe afin de récupérer une requête que je vais pouvoir modifier.

Je vais donc tenter de me connecter avec:

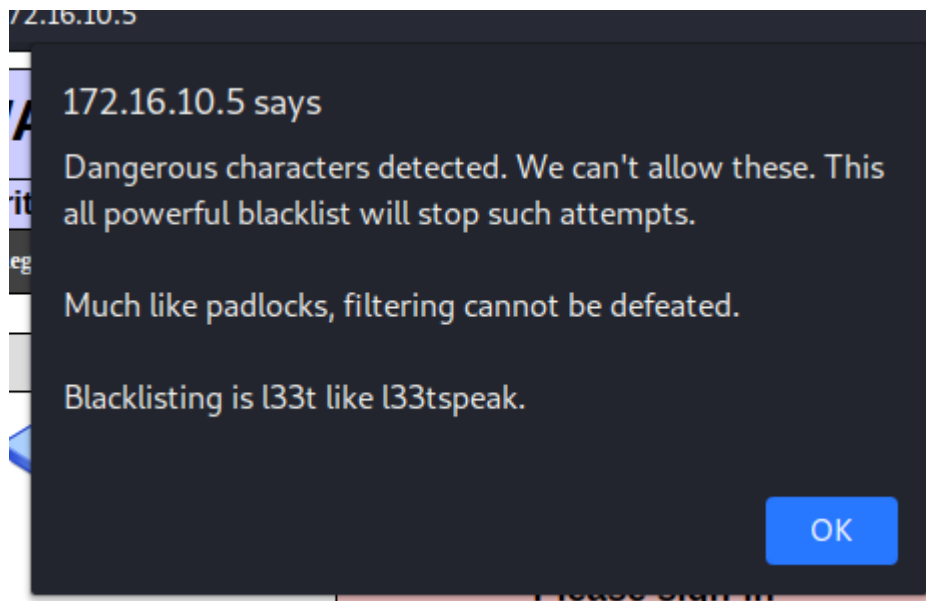
username : Matthias

Password : gfyzgf4157

Bien évidemment je sais que ce mot de passe n'est pas le bon mais on se met à la place de l'attaquant.



Lorsque que je clique sur login une erreur qui dit que je ne peux pas envoyer la requête car il détecte un élément dangereux. Burp Suite est blacklisté.



**D1.Q2.** Les informations affichées par le comparateur sont-elles exploitables pour tenter une énumération ?

Le navigateur nous dit qu'il détecte un élément dangereux. Qu'il ne peut pas autoriser cet élément qui est dans la liste noire. Donc nous savons que lorsque l'on met le niveau 5 de sécurité il y a donc plus de sécurité et un filtre qui bloque les attaques vers BurpSuite.

Par ailleurs, en inspectant la page du navigateur on trouve une fonction nommée onsubmit qui est sûrement liée au fait d'avoir le toggle security au level 5, l'erreur du navigateur provient de cela.

```
<script type="text/javascript">...</script>
<div style="margin: 5px;">...</div>
<script>...</script>
<div class="hint-wrapper-header" id="idHintWrapperHeader" title="Click to open this section" onclick="toggleBody(this, window.document.getElementById('idHintWrapperBody'), window.document.getElementById('idHintWrapperHeaderImage'));" onmouseover="this.style.backgroundColor='#cccccc';this.style.color='#ffffff';" onmouseout="this.style.backgroundColor='#FFFFFF';this.style.color='#000000';" style="display: block; background-color: rgb(255, 255, 255); color: rgb(0, 0, 0);">...</div>
<div id="idHintWrapperBody" class="hint-wrapper-body" style="display: none;">...</div>
<div id="id-log-in-form-div" style="text-align: center;">
  <form action="index.php?page=login.php" method="post" enctype="application/x-www-form-urlencoded" onsubmit="return onSubmitOfLoginForm(this);" id="idLoginForm">...</form> == $0
</div>
<table style="margin-left:auto; margin-right:auto;">...</table>
```

Cependant, on peut aller plus loin en recherchant la fonction “onSubmitOfLoginForm” avec Ctrl+f dans notre inspection de la page.

```
... >le.main-table-frame tbody tr td t
onSubmitOfLoginForm
```

Désormais on en connaît plus sur cette fonction, c’est donc du Js.

```
<script type="text/javascript">
<!--
  var l_loggedIn = false;
  var lAuthenticationAttemptResultFlag = -1;
  var lValidateInput = "FALSE"

  function onSubmitOfLoginForm(/*HTMLFormElement*/ theForm){
    try{
      if(lValidateInput == "TRUE"){
        var lUnsafeCharacters = /[`~!@#$$%^&*()-_+=\[\]
        {}\\|;':",./<?>?]/;
        if (theForm.username.value.length > 15 ||
            theForm.password.value.length > 15){
          alert('Username too long. We dont want to
allow too many characters.\n\nSomeone might have enough room to
enter a hack attempt.');
          return false;
        };// end if

        if
        (theForm.username.value.search(lUnsafeCharacters) > -1 ||
        theForm.password.value.search(lUnsafeCharacters) > -1){
          alert('Dangerous characters detected. We
can\'t allow these. This all powerful blacklist will stop such
attempts.\n\nMuch like padlocks, filtering cannot be
defeated.\n\nBlacklisting is l33t like l33tspeak.');
          return false;
        };// end if
      };// end if(lValidateInput)

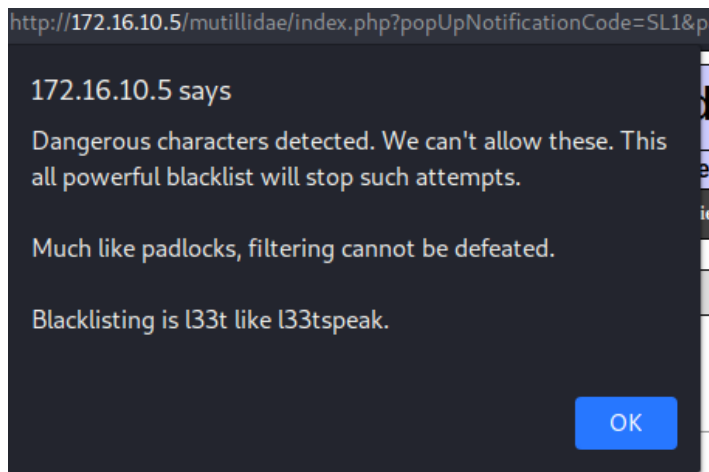
      return true;
    }catch(e){
      alert("Error: " + e.message);
    };// end catch
  };// end function onSubmitOfLoginForm(/*HTMLFormElement*/
  theForm)
  //-->
</script>
```

**D1.Q3.** Dans tous les cas, tentez une énumération et indiquez ce qui se passe (copies d’écran...).

#### Test level 1:

Je vais donc tenter de mettre le Toggle Security au niveau 1 et refaire le test entièrement afin de voir si on a toujours le message d’erreur.

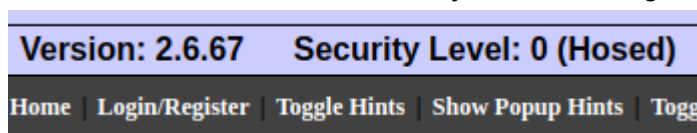




On a toujours notre message d'erreur, et la fonction est toujours présente. Je descends donc encore d'un niveau.

### Test level 0:

Encore une fois je vais donc mettre le Toggle Security au niveau 0 et refaire le test entièrement afin de voir si on a toujours le message d'erreur.



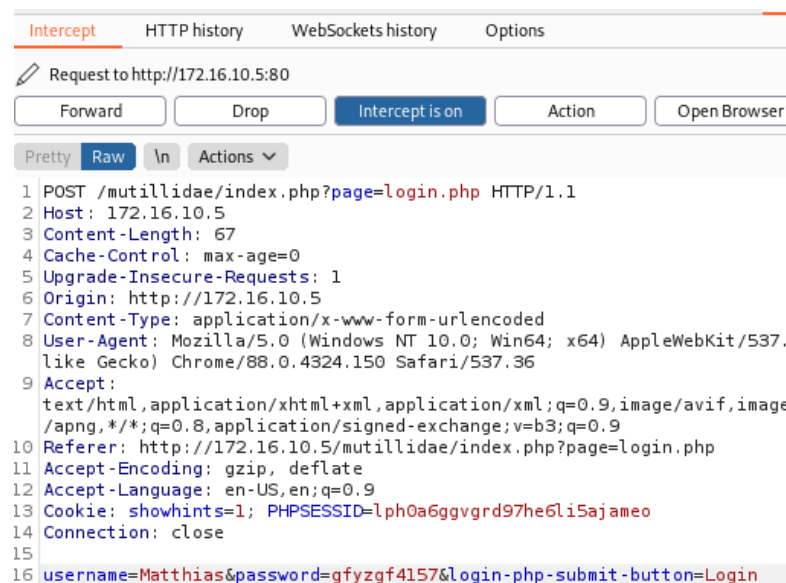
Par ailleurs, je rappelle mes identifiants de connexion choisis pour le test son:

username : Matthias

Password : gfyzgf4157 (mauvais mdp)

J'arrive à récupérer la requête dans le logiciel Burp Suite. On comprend donc que lorsque le niveau de sécurité est au plus faible le navigateur n'a plus le script en JS dans laquelle il y a la fonction "onSubmitOfLoginForm". Pour vérifier ceci j'inspecte la page et recherche la fonction. Que je ne trouve pas.

Requête BurpSuite:



Je peux donc continuer l'attaque:

Je clique droit sur la requête récupère et clique sur "Send to Intruder".

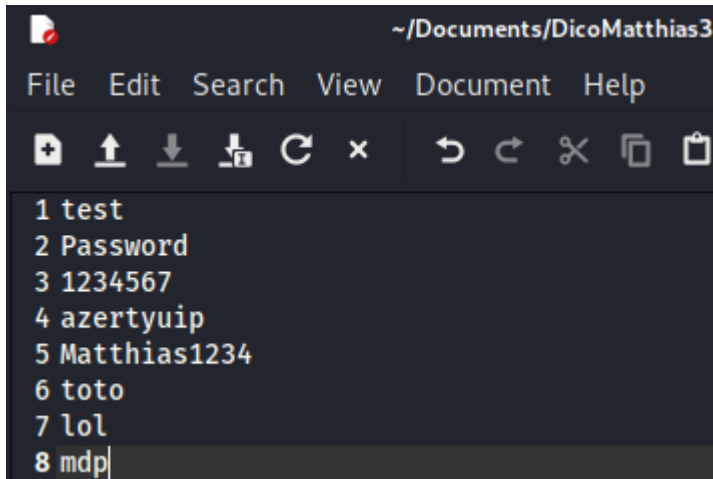
Je vais dans l'onglet Intruder et clique sur "Cleať §" afin de supprimer toutes les variables déjà créées.

Puis double clique sur mon faux password et Add §"

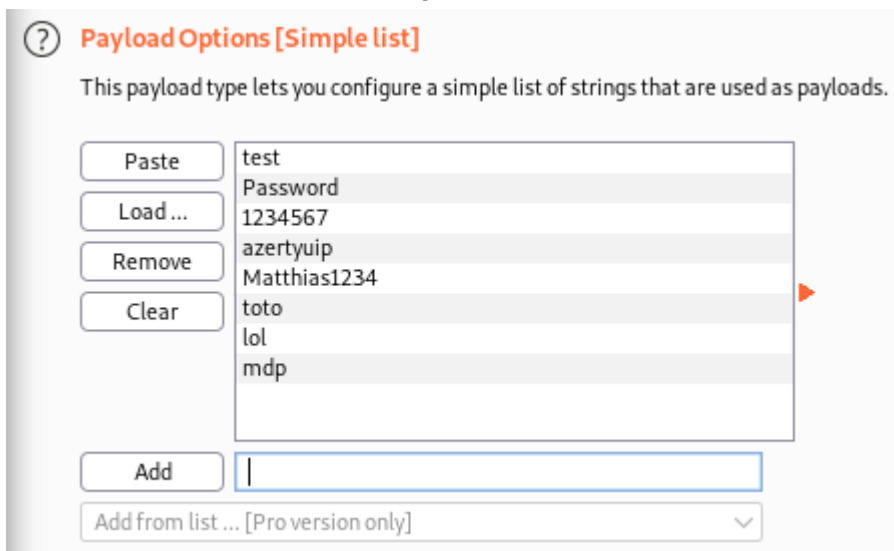
Voilà notre mot de passe est donc désormais en vert, c'est donc une variable:

```
username=Matthias&password=$gfyzgf4157$&login-php-submit-button=Login
```

Ensuite il faut créer notre propre dictionnaire de mot de passe. Avec le bon mot de passe dedans.



Puis on l'introduit dans le PayLoad ce qui va permettre de comparer tous les mot de passe. Et trouver le bon ou pas. En testant chacun des mots de passe requête par requête. Ce qui est donc facile à voir dans les logs du serveur mutillidae.



Puis cliquer sur "Start Attack" pour lancer l'attaque.

Voici la réponse de l'attaque que je viens de lancer. On peut y apercevoir qu'au niveau du mot de passe correct le Status est "302", ce qui veut dire que le password est correct. On l'a donc trouvé.

Intruder attack 1						
Attack Save Columns						
Results	Target	Positions	Payloads	Options		
Filter: Showing all items						
Request ^	Payload	Status	Error	Timeout	Length	
0		200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
1	test	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
2	Password	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
3	1234567	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
4	azertyuiop	200	<input type="checkbox"/>	<input type="checkbox"/>	55232	
5	Matthias1234	302	<input type="checkbox"/>	<input type="checkbox"/>	378	
6	toto	200	<input type="checkbox"/>	<input type="checkbox"/>	55320	
7	lol	200	<input type="checkbox"/>	<input type="checkbox"/>	55320	
8	mdp	200	<input type="checkbox"/>	<input type="checkbox"/>	55320	

Lorsque l'on clique sur la requête qui a fonctionné on peut voir son descriptif.

5	Matthias1234	302	<input type="checkbox"/>	<input type="checkbox"/>	378
---	--------------	-----	--------------------------	--------------------------	-----

Request Response

Pretty Raw In Actions

```
1 POST /mutillidae/index.php?page=login.php HTTP/1.1
2 Host: 172.16.10.5
3 Content-Length: 69
4 Cache-Control: max-age=0
5 Upgrade-Insecure-Requests: 1
6 Origin: http://172.16.10.5
7 Content-Type: application/x-www-form-urlencoded
8 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
  Safari/537.36
9 Accept:
  text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,application/javascript;v=b3;q=0.9
10 Referer: http://172.16.10.5/mutillidae/index.php?page=login.php
11 Accept-Encoding: gzip, deflate
12 Accept-Language: en-US,en;q=0.9
13 Cookie: showhints=1; PHPSESSID=lph0a6ggvgrd97he6li5ajameo
14 Connection: close
15
16 username=Matthias&password=Matthias1234&login-php-submit-button=Login
```

Mais aussi la réponse:

Request Response

Pretty Raw Render In Actions

```
1 HTTP/1.1 302 Found
2 Date: Tue, 29 Nov 2022 13:00:44 GMT
3 Server: Apache/2.4.29 (Ubuntu)
4 Expires: Thu, 19 Nov 1981 08:52:00 GMT
5 Cache-Control: no-store, no-cache, must-revalidate
6 Pragma: no-cache
7 Set-Cookie: username=Matthias
8 Set-Cookie: uid=137
9 Location: index.php?popUpNotificationCode=AUI
10 Content-Length: 0
11 Connection: close
12 Content-Type: text/html; charset=UTF-8
13
```

On peut en conclure que la méthode d'attaque par Force Brute fonctionne correctement quand le niveau de sécurité du serveur n'est pas très élevé.

**D1.Q4.** Chercher dans le code source de la page `ws-user-account.php` (située dans `/var/www/html/mutillidae/webservices/soap/`) le codage mis en place permettant d'obtenir un encodage sécurisé. Expliquer le rôle de l'instruction `EncodeforHTML`.

Aller dans la machine Mutillidae et chercher dans:

`/var/www/html/mutillidae/webservices/soap/`

ouvrir le fichier `ws-account.php`:

```
prof@prof:~$ cd /var/www/html/mutillidae/webservices/soap/
prof@prof:/var/www/html/mutillidae/webservices/soap$ ls
lib ws-hello-world.php ws-lookup-dns-record.php ws-user-account.php
prof@prof:/var/www/html/mutillidae/webservices/soap$ sudo nano ws-user-account.php
```

Pour trouver l'instruction plus facilement faire "Ctrl+W" et taper la recherche.

Voici donc l'instruction `EncodeforHTML`:

```
function doXMLEncodeQueryResults($pUsername, $pQueryResult, $pEncodeOutput){
    $lResults = "<accounts message=\"Results for {$pUsername}\">";
    $lUsername = "";
    $lSignature = "";
    while($row = $pQueryResult->fetch_object()){
        $pEncodeOutput?{$lSignature} = $lUsername = $Encoder->encodeForHTML
($row->username):$lUsername = $row->username;;
        if(isset($row->mysignature)){
            $pEncodeOutput?{$lSignature} = $Encoder->encodeForHTML
($row->mysignature):$lSignature = $row->mysignature;
        }// end if
        $lResults.= "<account>";
        $lResults.= "<username>{$lUsername}</username>";
        if(isset($row->mysignature)){ $lResults.=
"<signature>{$lSignature}</signature>";};
        $lResults.= "</account>";
    }// end while
}
```

Cette partie de code permet de créer une signature encodée lors de la première authentification(register) puis récupérer cette signature encodée et la met dans un tableau.. Puis après lors d'une connexion il compare la signature créer dans la tableau avec celle de l'utilisateur qui veut se connecter donc la nouvelle demande de connexion. Si la signature est correcte alors la connexion est établie mais si celle-ci est mauvaise alors l'utilisateur voulant se connecter ne pourra pas.

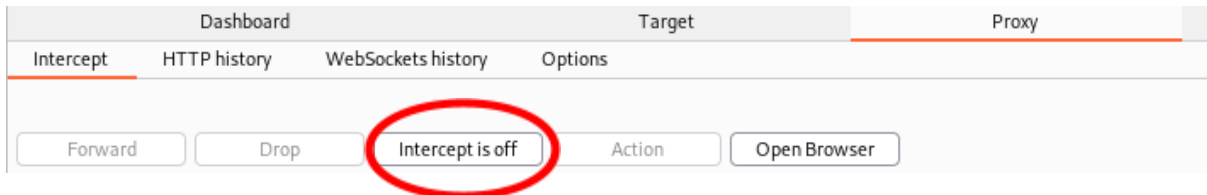
## APPRENTISSAGE 2 : vol de session

### Interception du cookie

L'objectif de cette partie est de trouver le login et le mot de passe de connexion d'un compte utilisateur en lui détournant les cookies d'authentification.

Pour commencer je vais me déconnecter de la mutillidae et arrêter la capture de trame du serveur proxy de BurpSuite.

Voilà comment arrêter la capture de trames, faire Proxy>Intercept>Intercept is off:



Puis je créer un compte totalement normal dans un nouveau navigateur. Que je note dans la description de ma VM.

**Please choose your username, password and signature**

**Username**

**Password**  [Password Generator](#)

**Confirm Password**

**Signature**

Note description VM:

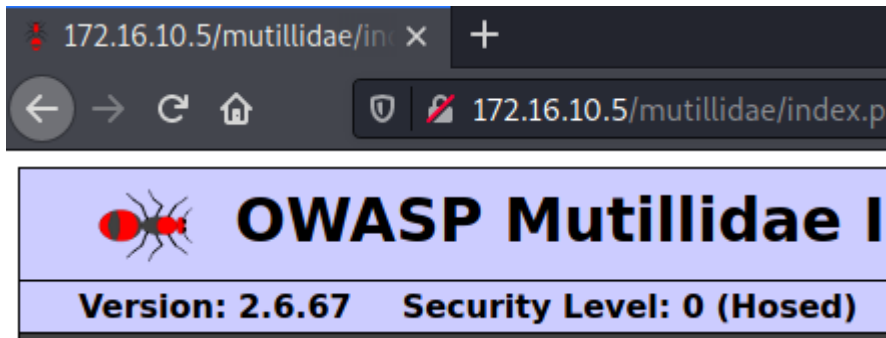
```
Compte Mutillidae:
Username: toto
Password:toto1234

Username: Matthias
Password: Matthias1234

Username: utilisateur1
Password: utilisateur1
|
```

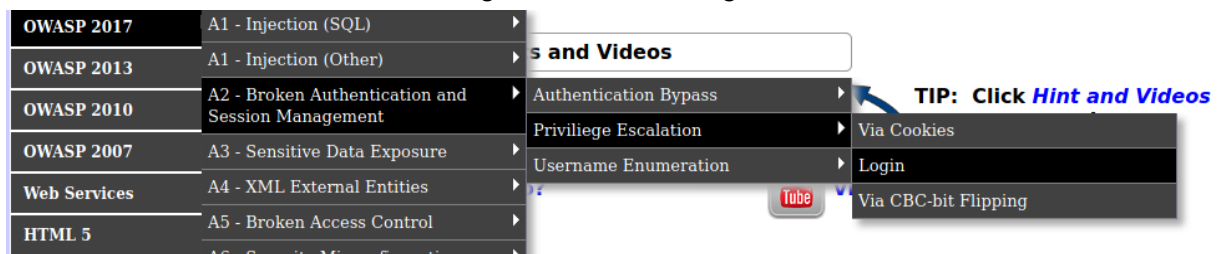
**A2.Q1** : Faites une copie d'écran qui montre ce niveau.

Puis mettre mutillidae en toggle security 0:



Retourner sur Burp Suite cliquer sur "Open Browser" et garder le "Intercept is off", on va réaliser une élévation des privilèges du compte que l'on vient de créer en se connectant au même moment.

Pour cela OWASP 2017> A2> Privilege Escalation > Login



Puis entrer l'authentification du compte toto puis cliquer sur "Login":

**Please sign-in**

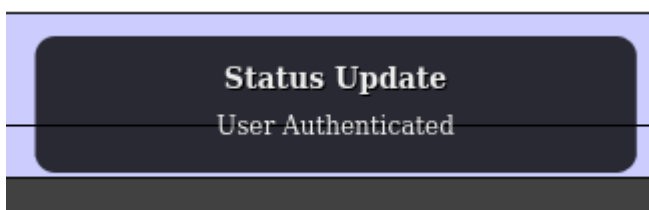
**Username**

**Password**

*Dont have an account? [Please register here](#)*

**A2.Q2** : Faites une copie d'écran du résultat obtenu et indiquez comment vous voyez que vous êtes bien connecté avec votre utilisateur.

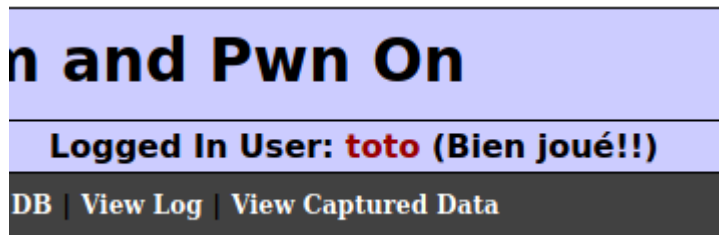
Ce message apparaît, on a donc bien plus de privilèges.





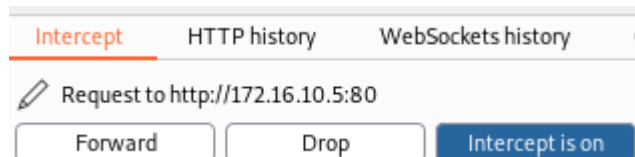
Je suis bien connecté en tant que toto.

On peut même voir la description de ma signature: "Bien joué!!"



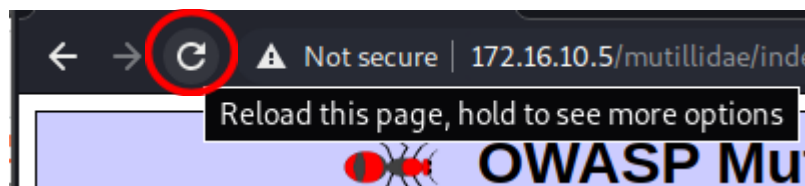
Ensuite on va donc autorisé le proxy Burp à faire des capture de trames:

Pour cela: "Intercept is on"



Puis actualiser la page afin de capter la requête:

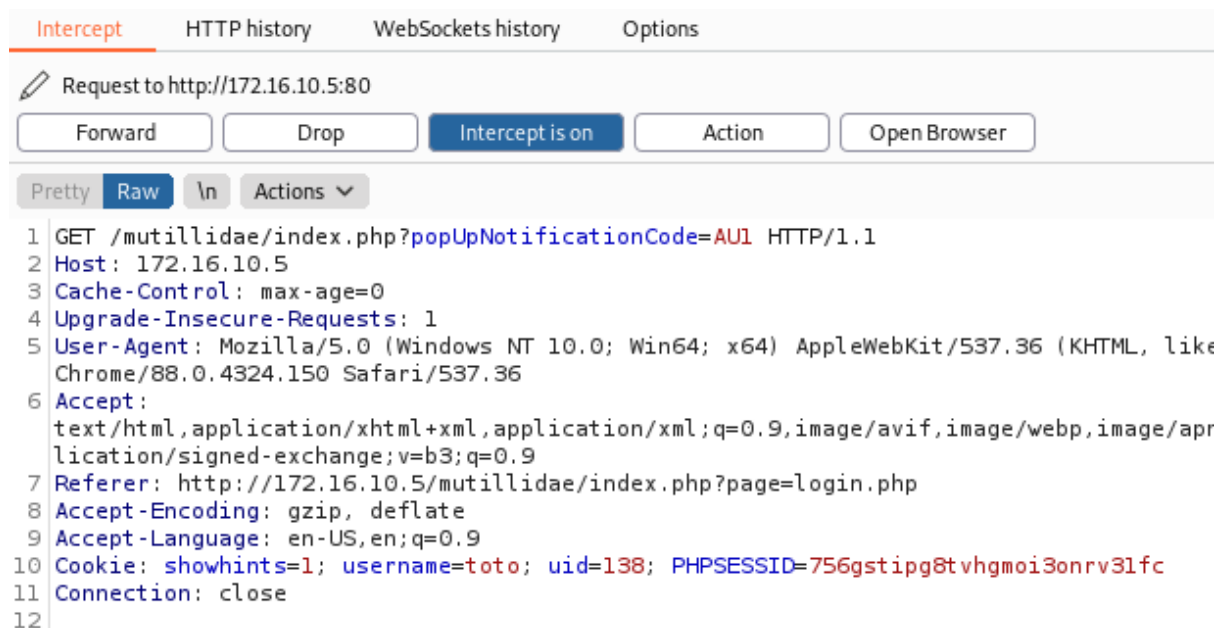
Pour cela:



ou "Ctrl + f5"

On sera donc automatiquement renvoyé sur la page Intercept du proxy BurpSuite.

Voilà donc la requête récupéré:



**A2.Q3** : Faites une copie d'écran pour montrer le cookie du nom de votre utilisateur. Quelle est sa valeur ?

Si on analyse mieux cette requête on peut donc voir différents cookies, on s'intéresse donc à 2 cookies en particulier: le nom de l'utilisateur "**username=toto**", le username id "**uid=138**"

```
Cookie: showhints=1; username=toto; uid=138; PHPSESSID=756gstipg8tvhgmoi3onrv31fc  
Connection: close
```

### Vol de la session

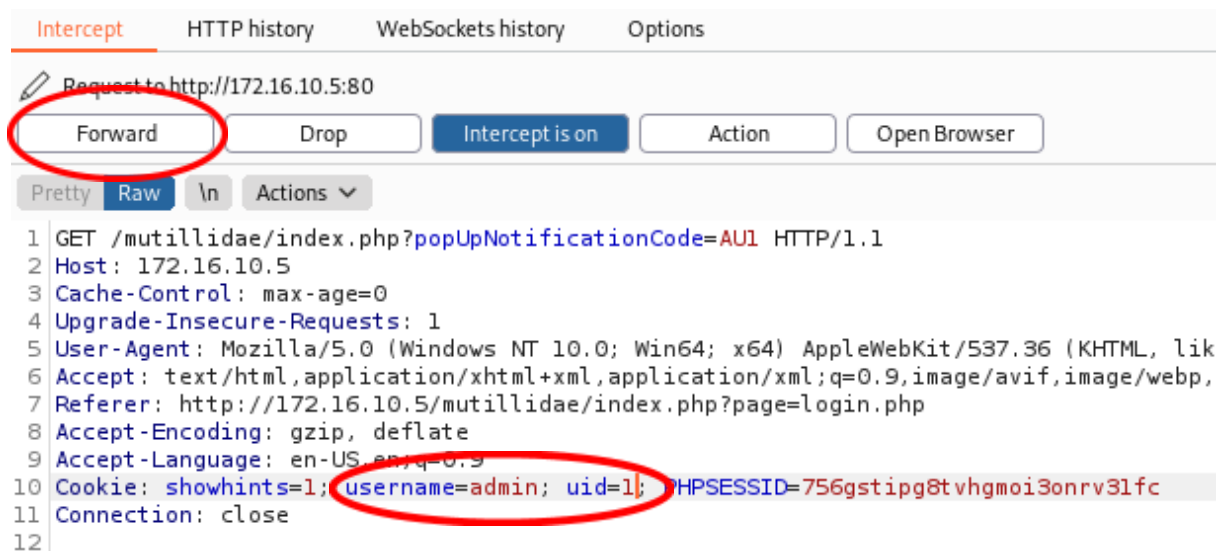
Dans cette partie on va tenter de voler la session admin grâce à l'uid de cookie récupéré dans la requête précédente

Pour débiter modifier la valeur de l'uid pas 1. Car on suppose que admin étant le premier utilisateur créé donc uid=1.

Saisir donc:

"username=admin"

"uid=1"



Intercept HTTP history WebSockets history Options

Request to http://172.16.10.5:80

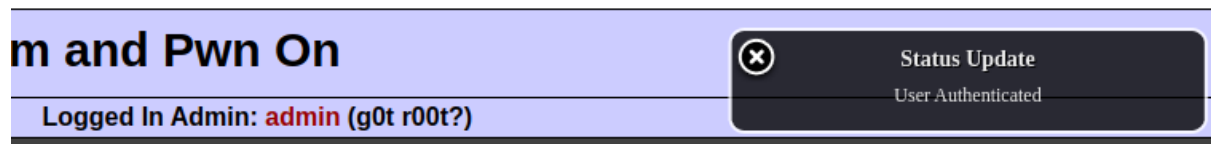
Forward Drop Intercept is on Action Open Browser

Pretty Raw \n Actions

```
1 GET /mutillidae/index.php?popUpNotificationCode=AU1 HTTP/1.1  
2 Host: 172.16.10.5  
3 Cache-Control: max-age=0  
4 Upgrade-Insecure-Requests: 1  
5 User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, lik  
6 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,  
7 Referer: http://172.16.10.5/mutillidae/index.php?page=login.php  
8 Accept-Encoding: gzip, deflate  
9 Accept-Language: en-US,en;q=0.9  
10 Cookie: showhints=1; username=admin; uid=1; PHPSESSID=756gstipg8tvhgmoi3onrv31fc  
11 Connection: close  
12
```

Puis cliquer sur "Forward":

On peut donc voir que l'on est connecté en tant qu' "admin".



m and Pwn On

Logged In Admin: admin (g0t r00t?)

Status Update  
User Authenticated

Nous arrivons donc à nous connecter avec le compte admin.

**A2.Q4** : Répondez à toutes ces questions :

Pourquoi a-t-on changé d'utilisateur ?

On a changé d'utilisateur pour se mettre en admin afin de pouvoir avoir les privilèges du compte administrateur. Pour pouvoir avoir accès à toutes les données.

Pourquoi sommes-nous maintenant connectés en tant qu'administrateur ?

Nous sommes connectés en tant qu'admin car nous avons récupéré une requête qui contenait les cookies de connexion et nous les avons modifiés. Par déduction nous avons choisi le uid=1 qui était le user admin. Par contre si nous avons augmenté le toggle sécurité de mutillidae au niveau 1 ou 5, nous n'aurions pas pu nous connecter avec ce uid.

Qu'elle est l'élément technique qui a permis ce changement, au niveau du serveur ?

L'élément technique qui a permis ce changement au niveau du serveur est que nous avons intercepté la trame envoyée vers le serveur depuis notre navigateur, puis l'avons modifiée afin de nous faire passer par l'utilisateur admin qui voulait se connecter.

S'agit-il d'une manipulation désirée par le serveur et légale ?

Cette manipulation n'est pas désirée par le serveur, elle n'est pas légale. C'est en quelque sorte une usurpation d'identité.

De plus, l'attaquant qui réalise cette attaque ne fait pas ça par hasard.

## CHALLENGE 2 : Codage sécurisé et analyse du code source

**D2.Q1.** Fermer puis relancer BurpSuite. Positionner le niveau de sécurité à 5 et relancer l'attaque en suivant les étapes 1 et 2. La modification du cookie uid a-t-elle une conséquence ?

Nous allons donc réessayer de faire une attaque de vol de session mais avec le toggle sécurité de mutillidae au niveau 5.

J'utiliserai le compte créé auparavant:

username: toto

password: toto1234

Je refait donc entièrement toute la manipulation:

Je vais sur Burpsuite désactiver le Interception des requêtes: "Intercept is off"

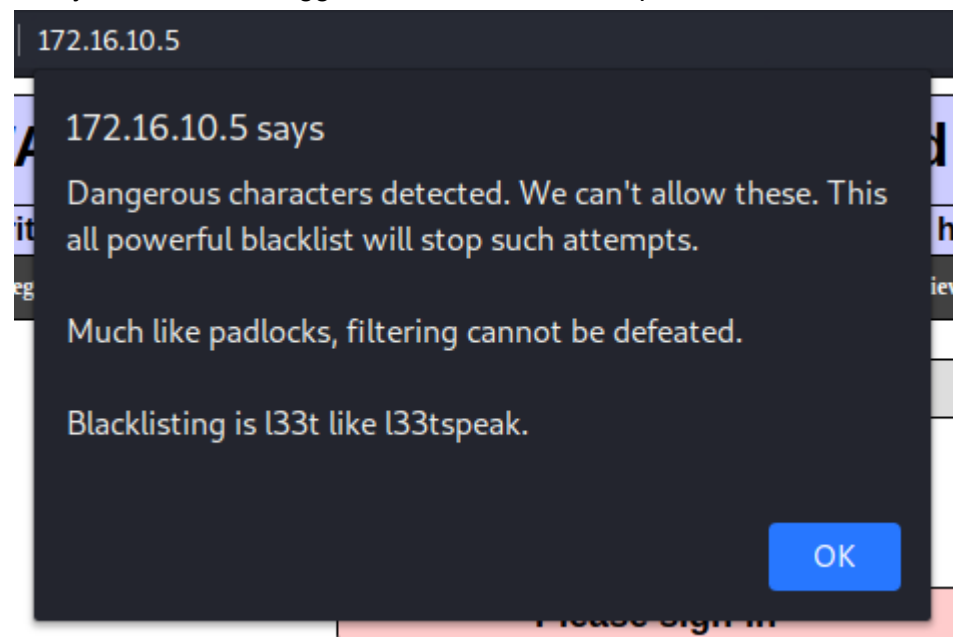
Puis "Open Browser", je tape l'url du serveur mutillidae. Je mets le toggle sécurité au level 5.

Puis OWASP 2017 > A2 > Privilege Escalation > Login

J'entre les identifiants de connexion du compte toto.

Un message d'erreur s'affiche:

Remarque: Il ressemble familièrement au message que nous avons déjà vu quand j'avais essayé de mettre le toggle sécurité au level 5 auparavant.



Nous n'arrivons donc pas à récupérer la requête car le navigateur nous bloque.

**D2.Q2.** Observez le code source de la page index.php (page d'accueil) et relever les différences avec le codage de niveau de sécurité 0 et 1.

level 0:

```

<html>
  <head>...</head>
  <body onload="onLoadOfBody(this);" == $0
    <table class="main-table-frame" border="1px" cellspacing="0px" cellpadding="0px">...
    </table>
    <!-- Bubble hints code -->
    <script type="text/javascript">...</script>
    <div style="border: 1px solid black;">...</div>
    <script type="text/javascript">...</script>
    <script type="text/javascript" src="javascript/jquery/jquery.js"></script>
    <script type="text/javascript" src="./javascript/jquery/jquery.balloon.js">
    </script>
    <script src="javascript/jquery/colorbox/jquery.colorbox-min.js"></script>
    <link rel="stylesheet" href="javascript/jquery/colorbox/colorbox.css">
    <script>...</script>
    <div id="cboxOverlay" style="display: none;"></div>
    <div id="colorbox" class style="display: none; padding-bottom: 42px; padding-right:
    42px;">...</div>
  </body>
</html>

```

level 1:

```

<html>
  <head>...</head>
  <body onload="onLoadOfBody(this);" == $0
    <table class="main-table-frame" border="1px" cellspacing="0px" cellpadding="0px">...
    </table>
    <!-- Bubble hints code -->
    <script type="text/javascript">...</script>
    <div style="border: 1px solid black;">...</div>
    <script type="text/javascript">...</script>
    <script type="text/javascript" src="javascript/jquery/jquery.js"></script>
    <script type="text/javascript" src="./javascript/jquery/jquery.balloon.js">
    </script>
    <script src="javascript/jquery/colorbox/jquery.colorbox-min.js"></script>
    <link rel="stylesheet" href="javascript/jquery/colorbox/colorbox.css">
    <script>...</script>
    <link rel="stylesheet" type="text/css" href="styles/gritter/jquery.gritter.css">
    <script type="text/javascript" src="javascript/gritter/jquery.gritter.min.js">
    </script>
    <script>...</script>
    <div id="cboxOverlay" style="display: none;"></div>
    <div id="colorbox" class style="display: none; padding-bottom: 42px; padding-right:
    42px;">...</div>
  </body>
</html>

```

On peut voir que de nouveau script ce sont ajouté comme par exemple "onSubmitOfLoginForm" et d'autres qui permettent une meilleur sécurité.

**D2.Q3.** Expliquer les différences entre un cookie et une session. Conclure sur les bonnes pratiques de codage concernant le suivi des utilisateurs identifiés.

Tout d'abord les cookies et les sessions contiennent des informations sur l'utilisateur. Mais les différences sont que les cookies sont stockés côté client, navigateur web tandis que les sessions sont stockées côté serveur pour nous c'était mutillidae.

Les bonnes pratique en matières de codage concernant le suivi des utilisateurs sont:

- Il est préférable d'utiliser des valeurs uid générées de manières aléatoires et non prévisibles;
- Il faut qu'à chaque authentification avec succès l'id de session soit modifié;
- Et pour finir encoder les id de sessions.