TAMDA Zohir
BTS SIO 2 option SLAM
Bloc 3 - Cybersécurité

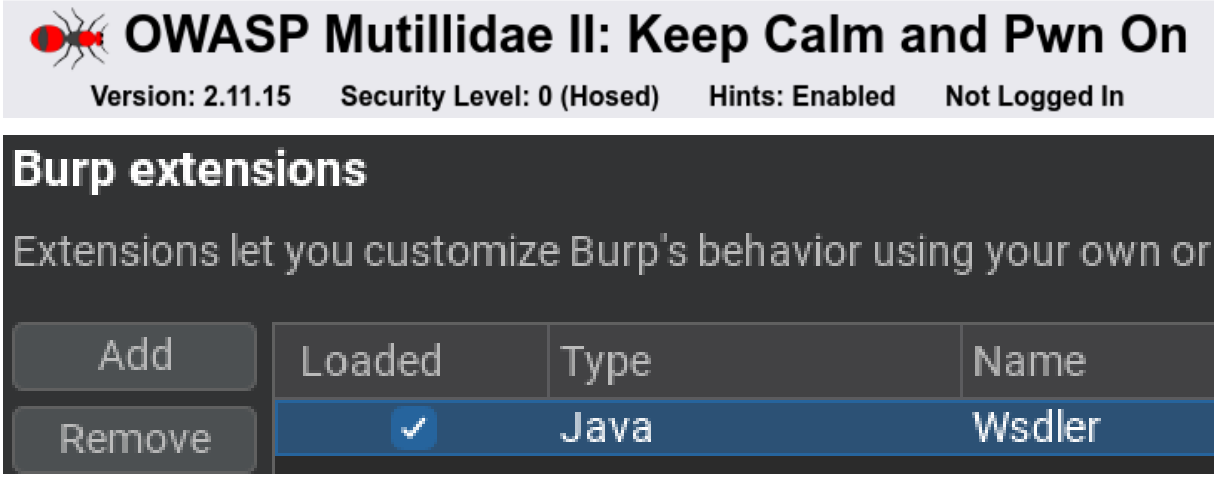
Le 24 septembre 2024

Enumération des logins

Sommaire

Mode non-sécurisé

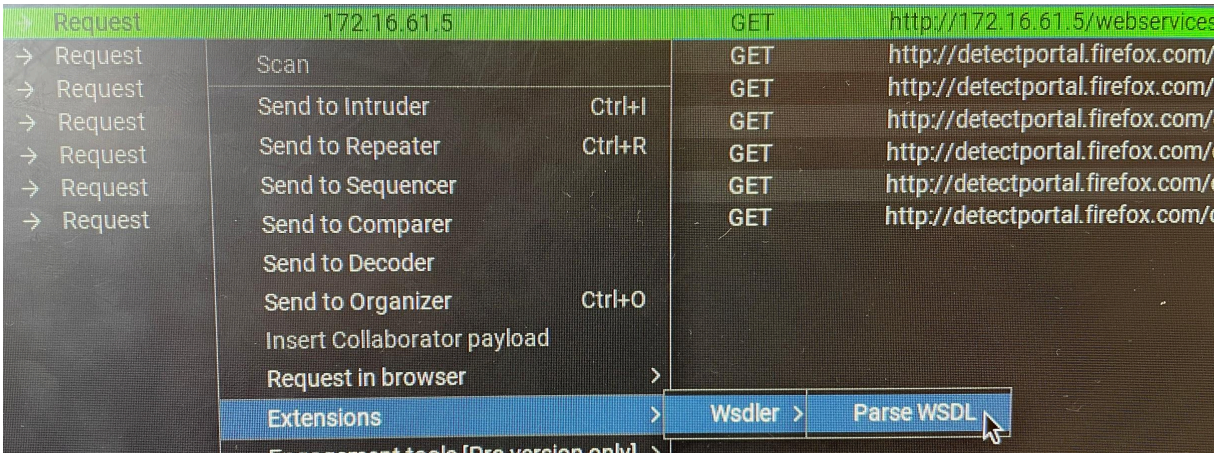
*Le niveau de sécurité est définie à "**0**", ce qui signifie qu'il n'y a aucune vérification à faire côté client ni côté serveur. L'extension "**Wsdler**" va être utilisée pour intercepter des requêtes WSDL.*

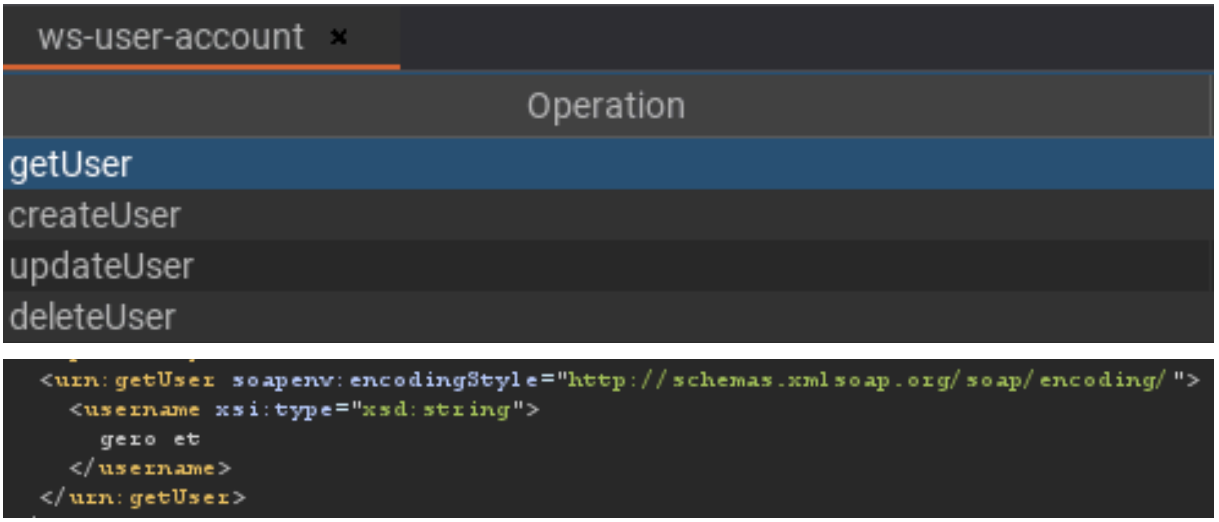


Vérification d'un utilisateur inexistant

L'URL "http://172.16.61.5/webservices/soap/ws-user-account.php?wsdl" a été tapée dans le navigateur pour permettre de récupérer des opérations dans le WSDL, l'interception est activé du côté de BurpSuite.

Pour récupérer les opérations, on peut parser le WSDL en cliquant sur "Parse WSDL" à partir du menu.



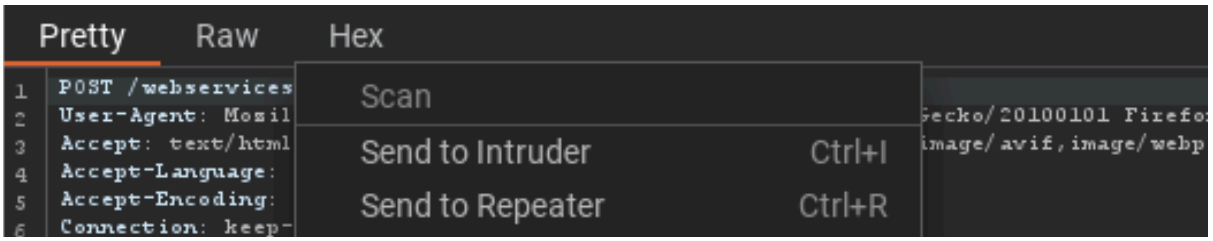


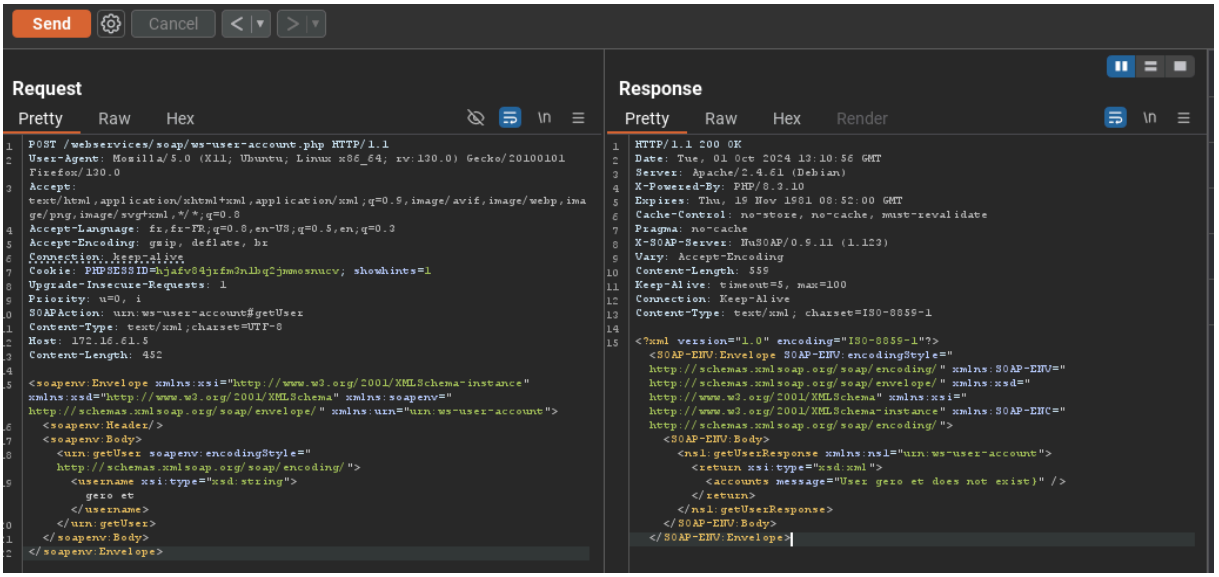
Résultat : on obtient 4 opérations :

- **`getUser`** qui sert à récupérer les informations de l'utilisateur.
- **`createUser`** qui sert à créer un utilisateur.
- **`updateUser`** qui sert à mettre à jour les informations de l'utilisateur.
- **`deleteUser`** qui sert à supprimer l'utilisateur.

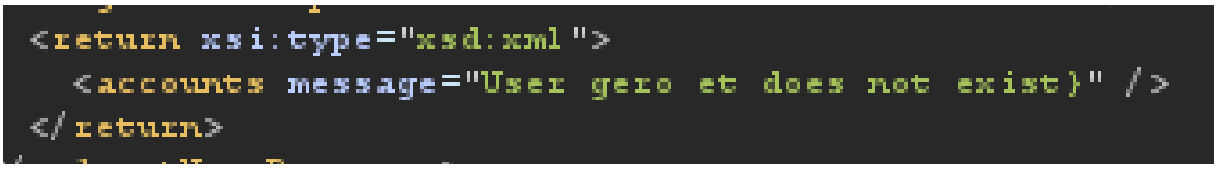
En dessous de cette liste des opérations, on obtient un code sous format XML qui contient la balise "**`username`**" et cette dernière contient une valeur qui correspond au nom de l'utilisateur "**`gero et`**" par exemple.

Pour cela, faire un clic droit sur le code et cliquer sur "Send to Repeater" *(Envoyer au répéteur)*.



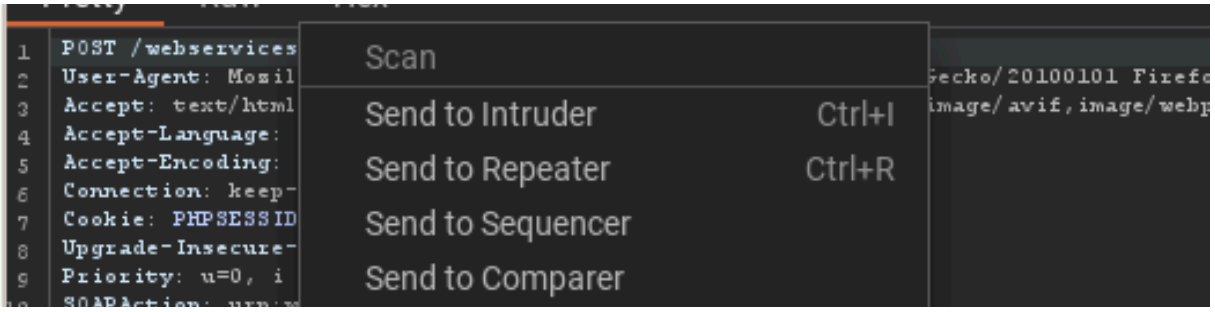


Le répéteur nous permet de modifier le code XML afin d'écrire le nom d'utilisateur et ainsi vérifier s'il existe ou pas.
Dans le cas où l'utilisateur "**gero et**" n'existe pas dans la base de données, la réponse sous format XML est claire: l'utilisateur gero et n'existe pas.



La dernière étape qui est indispensable consiste à comparer les données pour chaque requête XML qu'on effectue. Par exemple, on peut comparer la réponse avec un utilisateur inexistant et l'autre avec l'utilisateur existant. C'est le rôle du comparateur *(Comparer)*.

Pour cela, faire un clic-droit sur la réponse du code XML et cliquer sur "Send to Comparer" (Envoyer au comparateur).



Les deux requêtes vont être comparées dans la prochaine page.

Vérification d’un utilisateur existant

*On souhaite vérifier que l’utilisateur nommé “**utilisateur1**” existe bien dans la base de données. Les procédures pour récupérer les opérations WSDL, mettre la requête dans le répéteur ainsi que mettre la réponse dans le comparateur sont les mêmes.*

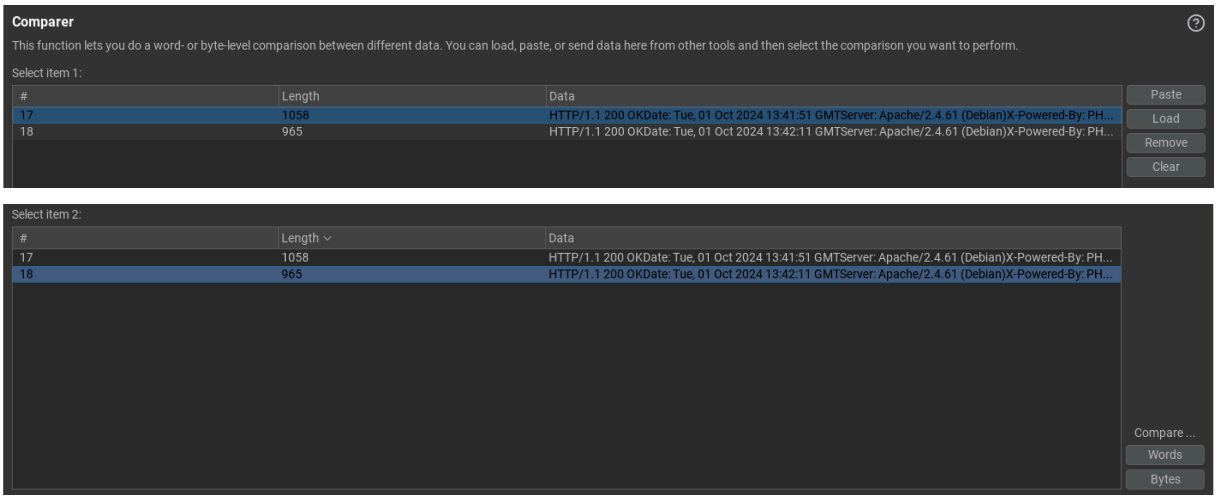
Si on veut envoyer la requête, cliquer sur le bouton “Send”.

```
<urn:getUser soapenv:encodingStyle="
http://schemas.xmlsoap.org/soap/encoding/">
   <username xsi:type="xsd:string">
      utilisateur1
   </username>
</urn:getUser>
```

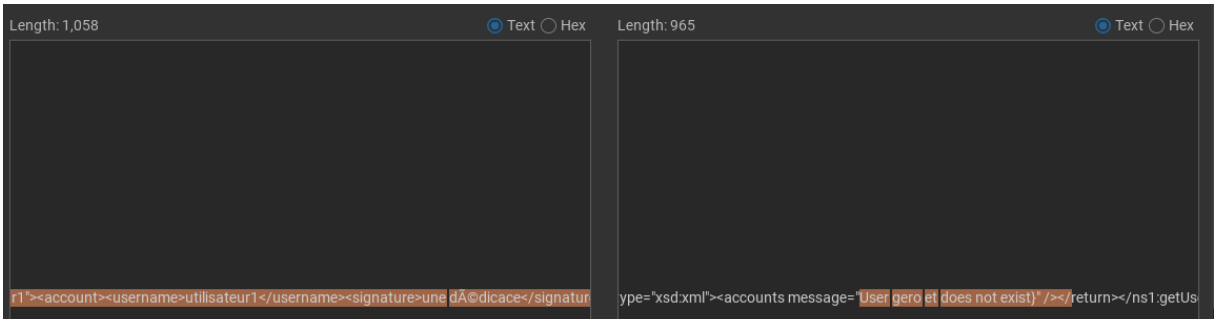
```
<accounts message="Results for utilisateur1">
  <account>
    <username>
      utilisateur1
    </username>
    <signature>
      une dÃ©dicace
    </signature>
  </account>
</accounts>
```

Requête *Réponse*

Résultat : on obtient le nom d’utilisateur ainsi que sa signature “Une dédicace”. Cette image démontre que l’utilisateur nommé “**`utilisateur1`**” existe.
Côté comparateur, il y a deux requêtes. Avec ça, on peut les comparer grâce au bouton “**`Words`**” qui compare chaque mot pour ces deux requêtes.

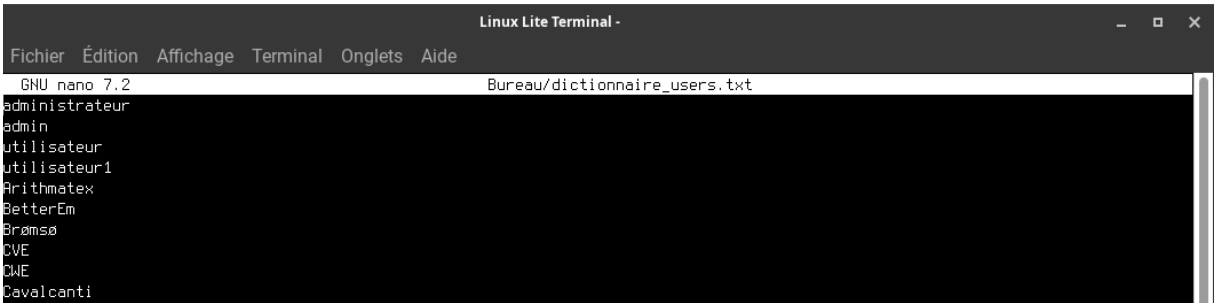


A gauche, il y a la réponse sous format XML contenant le nom d’utilisateur “**`utilisateur1`**” ainsi que la signature “**`Une dédicace`**” surlignés en orange; tandis qu’à droite, il y a également la réponse sous le même format mais ce message qui est surligné en orange indique que l’utilisateur “**`gero et`**” n’existe pas dans la base de données.

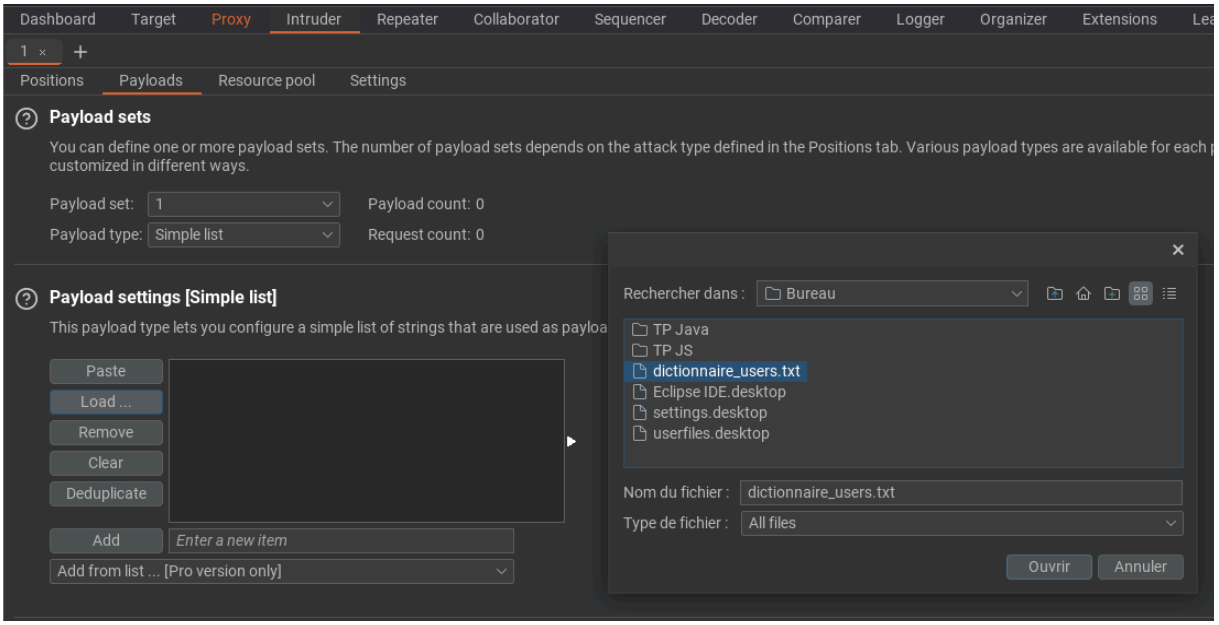


L'attaque brute force

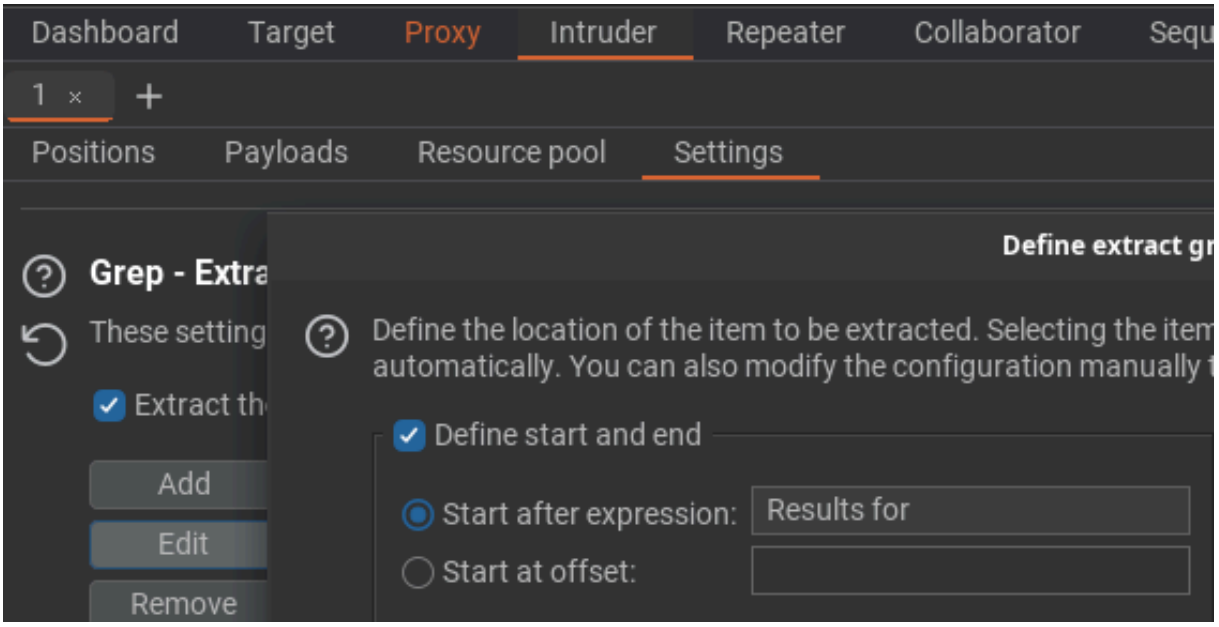
Pour réaliser cette attaque, il est nécessaire de créer un fichier texte contenant un ensemble de noms d'utilisateurs.



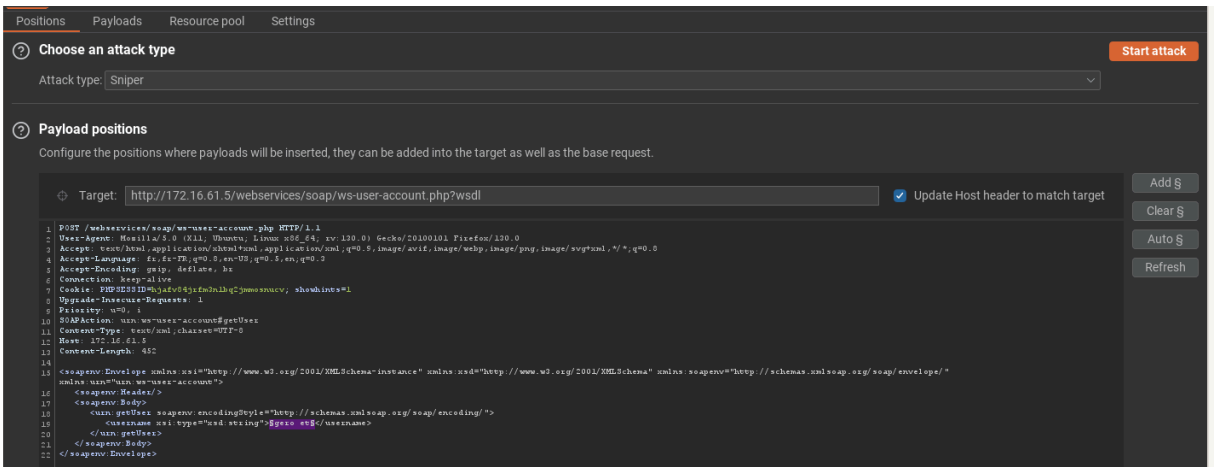
Pour charger ces utilisateurs, aller dans Intruder > Payloads et cliquer sur "Load...". Puis, charger le fichier texte contenant les noms d'utilisateurs, "dictionnaire_users.txt" par exemple



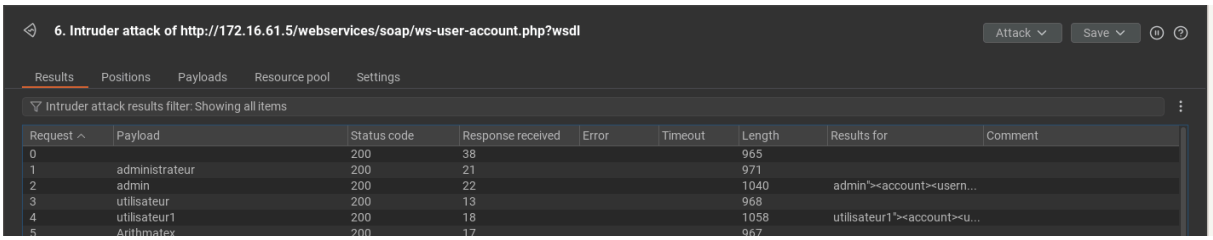
Puis, on ne souhaite afficher que les utilisateurs existants dans la base de données; il faut les filtrer. Pour cela, aller dans l'onglet "Settings", se rendre dans la section "Grep - Extract" et cliquer sur le bouton "Add" *(Ajouter)*. Puis, écrire dans le 1er champ de texte "Results for". Ce filtre permet de récupérer des données contenant le texte "Results for", s'il y en a pas, elles ne seront pas prises en compte.



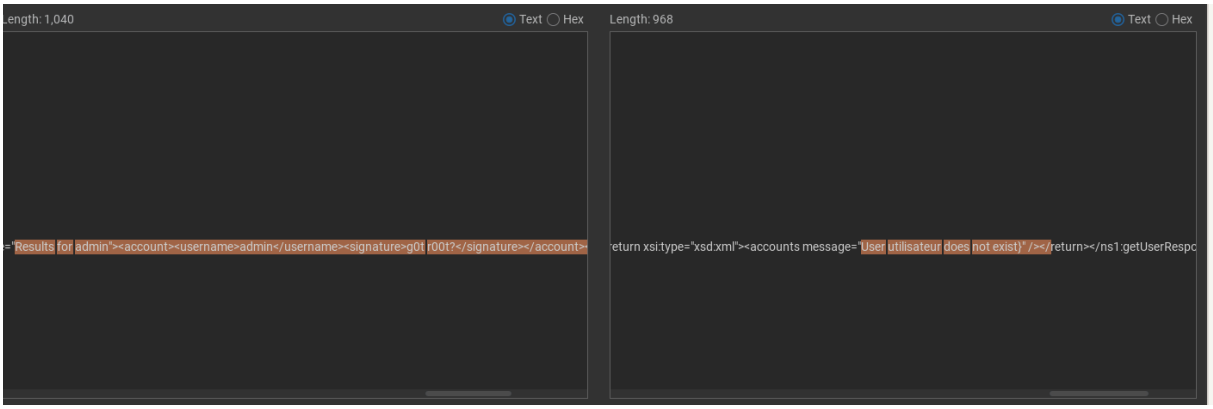
Après cela, dans l'onglet "Positions", ajouter l'URL concernée et coller le code XML.



Après ça, sélectionner "**`gero et`**" dans ce code et cliquer sur le bouton "**Add §**". Cela permet de remplacer "**`gero et`**" par chaque nom de l'utilisateur, ce qui est surligné en violet. Par la suite, on peut effectuer cette attaque en cliquant sur le bouton "Start attack".



Résultat, on voit que 2 informations sont présentes dans la colonne “Results for”, ce qui signifie que les utilisateurs “**`admin`**” et “**`utilisateur1`**” existent dans la base de données; tandis que d’autres qui n’ont aucune information dans cette colonne n'existent pas.



Grâce au comparateur, on peut voir en orange qu’avec le nom d’utilisateur et la signature présent dans cette ligne, l’attaquant a pu avoir ces informations pour pirater un compte utilisateur.

Mode sécurisé

Le niveau de sécurité est défini à 5. Les procédures sont identiques pour réaliser l'attaque brute force.

Les informations, à savoir le nom d'utilisateur ainsi que la signature peuvent toujours être exploitables mais on constate qu'au niveau de la signature, certains caractères spéciaux sont codés, ce qui fait que l'on obtient pas en clair.

```
function doXMLEncodeQueryResults($pUsername, $pQueryResult, $pEncodeOutput){

    global $Encoder;

        $lResults = "<accounts message=\"Results for {$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

        $lResults.= "</accounts>";

        return $lResults;

}//end function doXMLEncodeQueryResults
```

Après analyse du code dans le fichier "**ws-user-account.php**", la fonction **doXMLEncodeQueryResults()** code les résultats provenant des requêtes SQL avant le l'envoi de la réponse sous format XML. Grâce à la variable globale **$Encoder**, on peut coder les résultats en appelant la fonction **encodeForHTML()** à partir de cette variable. Cette fonction est utilisée pour coder le nom d'utilisateur ainsi que la signature.