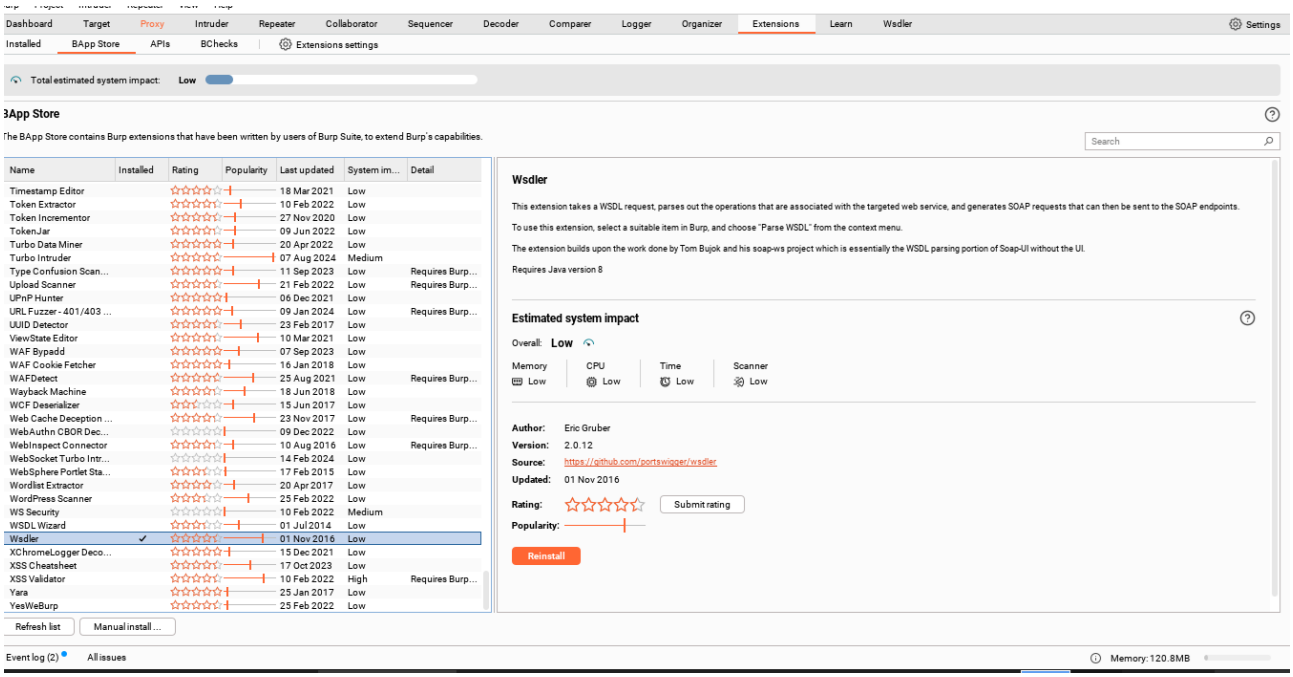
Ayoub
Hassani

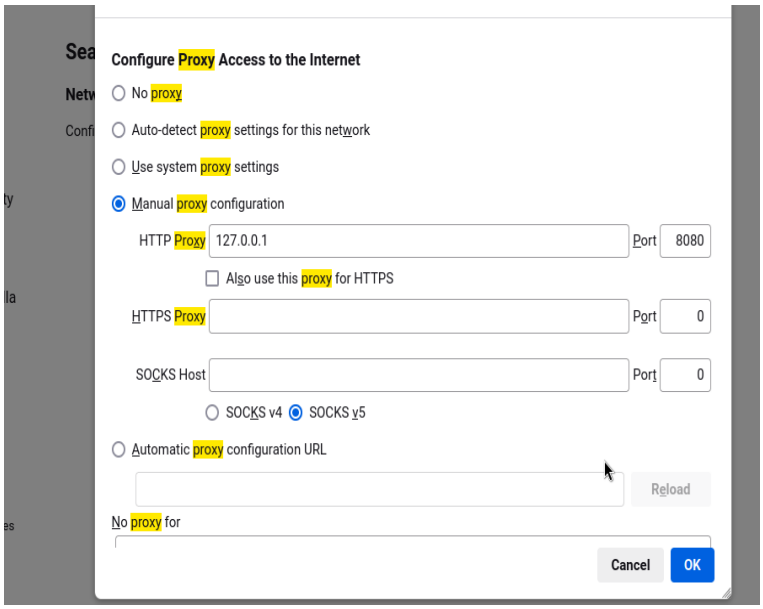
Compte rendu

Q1) Commencer par installer l'extension Wsdler en réalisant les manipulations décrites dans l'étape n°1. Puis, positionner le niveau de sécurité à 0. :

- L'extension Wsdler a été installée dans BurpSuite via le Bapp Store. Cette extension permet d'intercepter les requêtes SOAP et de manipuler les fichiers WSDL, qui sont des descriptions des services web disponibles

-Le proxy de BurpSuite a été configuré pour intercepter les requêtes HTTP du navigateur Firefox. Les requêtes et réponses de l'application web sont redirigées à travers BurpSuite.



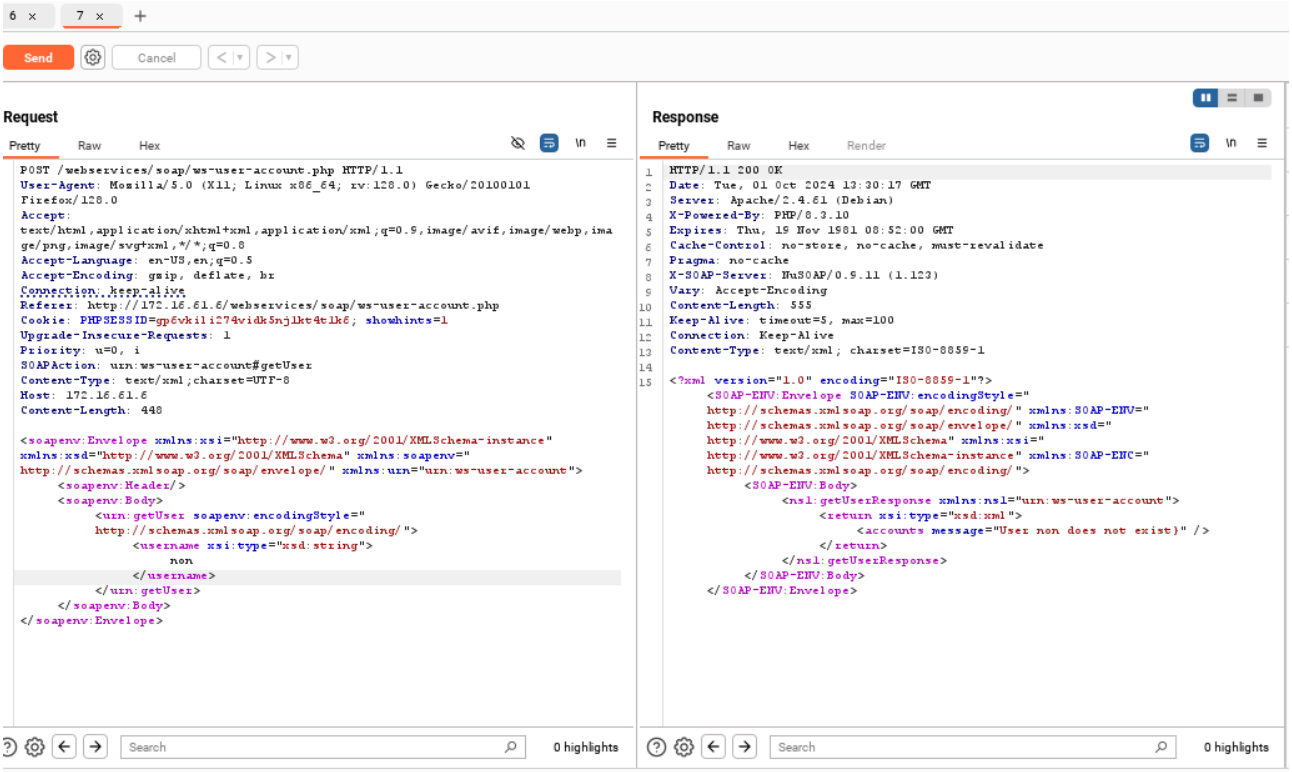


Q2) Tester un exemple de requête et de réponse à l'aide d'un login non valide en réalisant les manipulations décrites dans l'étape n°2 (parse de la page wsdl, envoi au répéteur, génération de la réponse et envoi au comparateur) :

L'application Mutillidae a été configurée pour tester l'énumération de logins. En accédant au service web SOAP proposé par l'application, la requête SOAP a été modifiée pour utiliser un login non valide, ici « gero et ».

-La requête a été envoyée au serveur, et la réponse obtenue est visible dans BurpSuite : "User gero et does not exist". Cette réponse confirme que le serveur renvoie un message explicite lorsqu'un login non valide est utilisé.

-Cette réponse a été envoyée au Comparer de BurpSuite, qui sera utilisé plus tard pour analyser les différences avec une réponse pour un login valide.

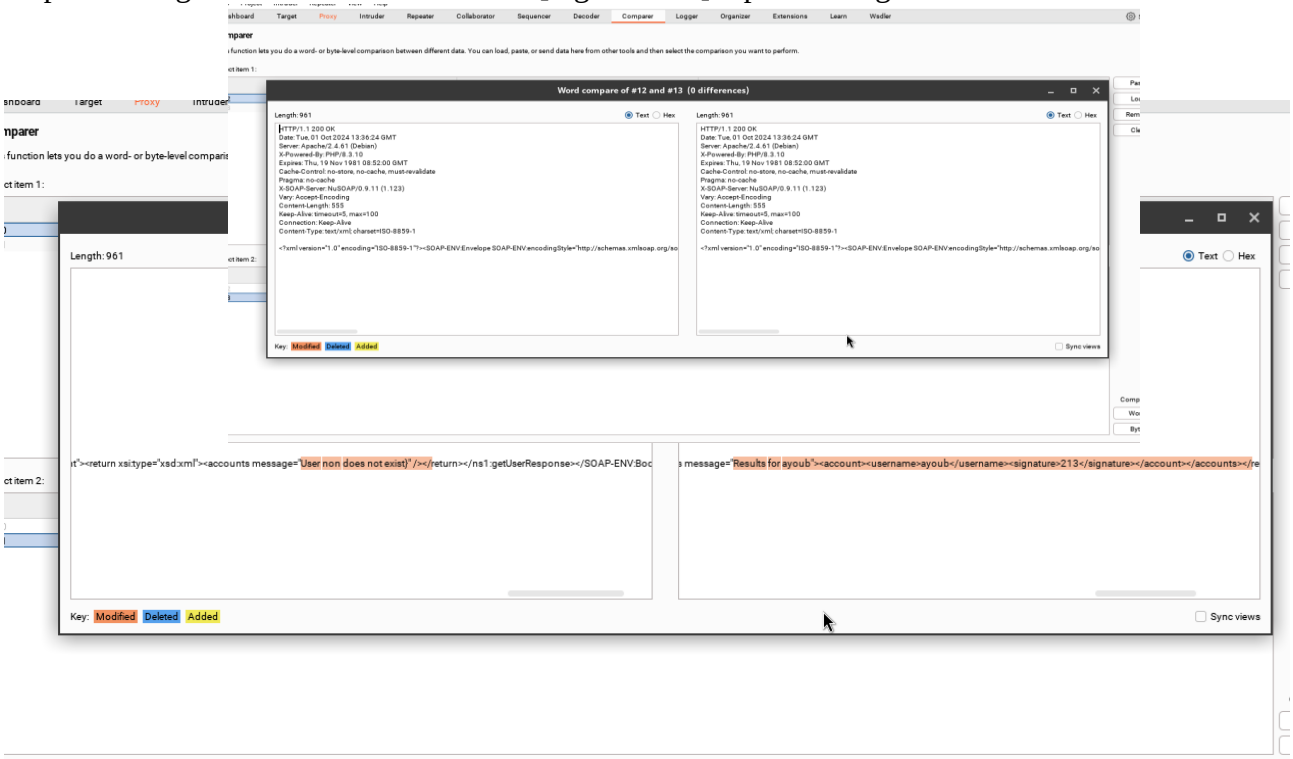


Q3) Tester un exemple de requête et de réponse à l'aide d'un login valide en réalisant les manipulations décrites dans l'étape n°3 (parse de la page wsdl, envoi au répéteur, modification avec un login valide, génération de la réponse et envoi au comparateur) :

Un login valide a été utilisé pour tester l'application. Dans ce cas, le login ayoub a été testé. La requête SOAP a été envoyée avec ce login, et la réponse renvoyée par le serveur indique un succès : "Results for ayoub".

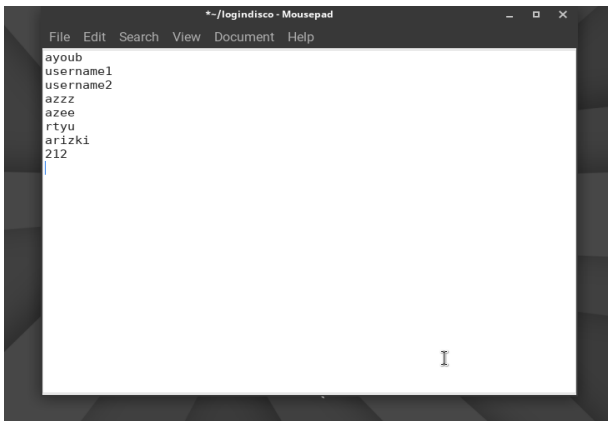
-Cette réponse a été également envoyée au Comparer de BurpSuite pour comparaison avec celle obtenue dans la question 2.

-Les réponses avec un login valide et un login non valide ont des différences notables. Le Comparer de BurpSuite met en évidence les lignes spécifiques, notamment les messages « User does not exist » pour un login invalide et « Results for [login valide] » pour un login valide.



Q4) Créer un dictionnaire de login sur votre machine cliente. Pour cela, ouvrir un éditeur de texte et saisir des logins les uns en dessous des autres et enregistrer votre fichier :

-Un fichier texte a été créé avec différents logins potentiels (par exemple, ayoub, username1, username2, etc.). Chaque login a été placé sur une ligne distincte dans un éditeur de texte.



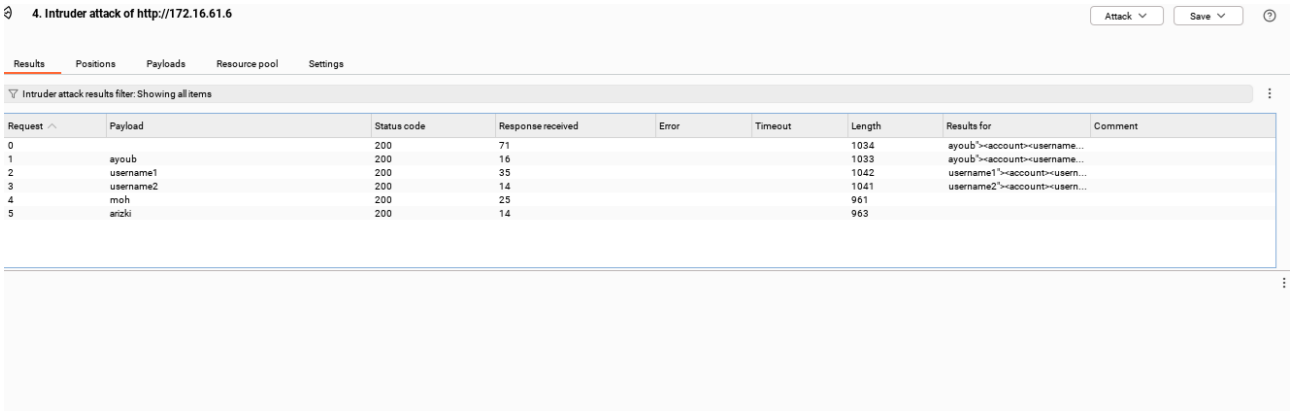
Q5)Lancer l'énumération et relever les logins valides en réalisant les manipulations décrites dans l'étape n°4.

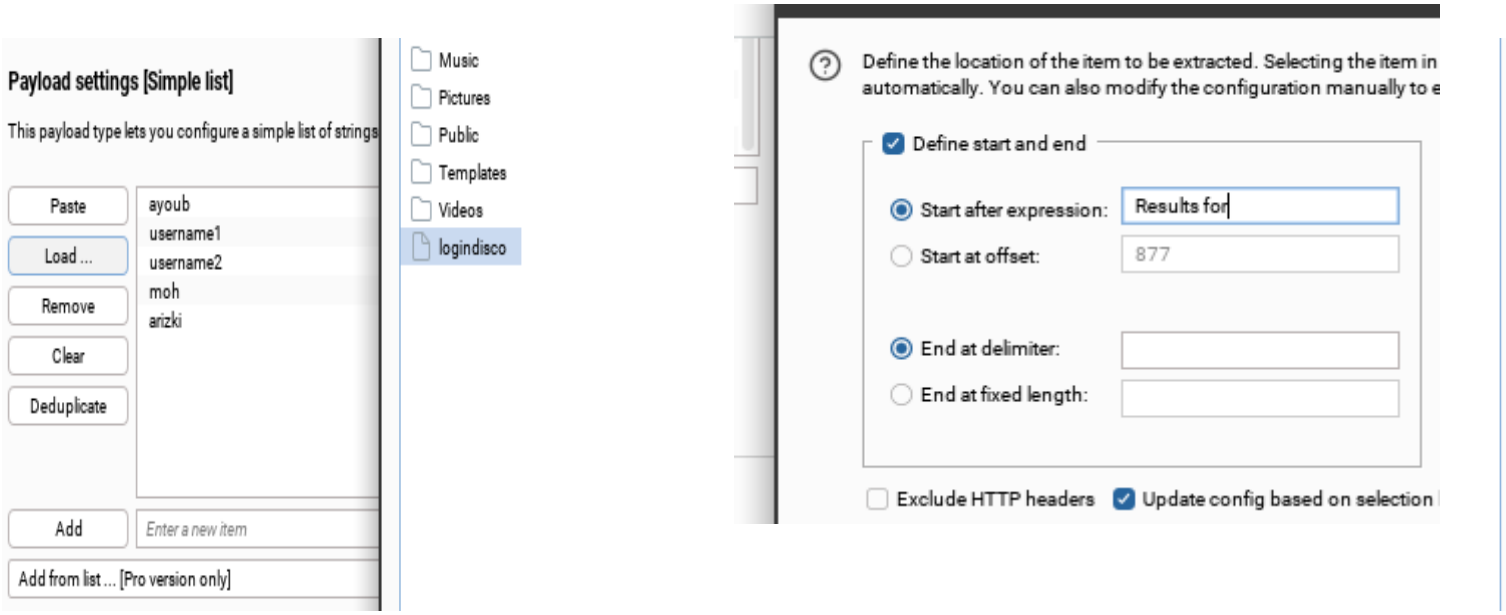
La requête SOAP a été envoyée au module Intruder de BurpSuite pour automatiser l'énumération des logins à partir du fichier dictionnaire créé en Q4.

-Dans l’onglet Payload, le fichier dictionnaire a été chargé, et dans l’onglet Positions, le champ du login a été marqué comme variable à remplacer lors de l'attaque.

-Un filtre a été ajouté pour repérer les réponses avec le texte « Results for », permettant ainsi de détecter automatiquement les logins valides.

-L’attaque a été lancée, et les résultats montrent que les logins « ayoub », « username1 », et « username2 » sont valides.





Q6) A l’aide du comparateur, expliquer quelles sont les lignes de la réponse sur lesquelles l’attaquant a pu s’appuyer pour lancer l’attaque ?

L’outil Comparer de BurpSuite a été utilisé pour comparer les réponses du serveur entre un login valide et un login non valide :

-Les lignes pertinentes sont celles qui diffèrent dans les deux réponses. Pour un login invalide, le message est « User does not exist », tandis que pour un login valide, on obtient « Results for [login] ».

-Cette différence dans les réponses permet à un attaquant de déterminer facilement quels logins sont valides dans l'application.

Travail à faire 2 :

Q1)Fermer puis relancer BurpSuite. Positionner le niveau de sécurité à 5 et relancer l’attaque en suivant les étapes 2 à 4:

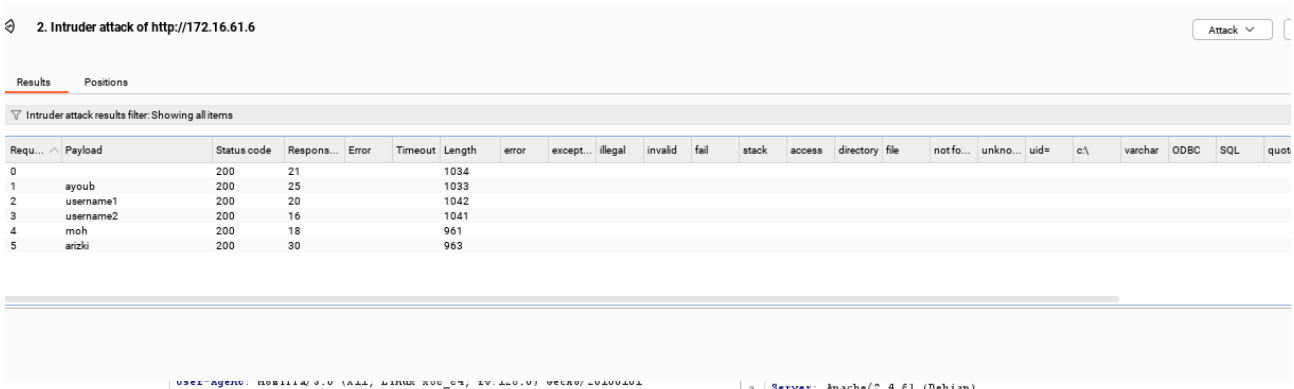
-Redémarrage de BurpSuite : Après avoir testé l'application en mode non sécurisé, BurpSuite a été relancé et reconfiguré pour l'attaque en mode sécurisé.

-Changement du niveau de sécurité : L'application web Mutillidae a été configurée avec un niveau de sécurité de 5, ce qui active les protections contre les attaques par énumération de logins.

- ❿ Lancement de l’attaque avec le mode sécurisé :

-L’attaque précédente a été relancée en suivant les mêmes étapes que dans le Travail à faire 1, c’est-à-dire en interceptant les requêtes SOAP et en utilisant un dictionnaire de logins pour tester les réponses.

-Les requêtes SOAP ont été envoyées en utilisant à nouveau le service getUser de l'API SOAP de Mutillidae.



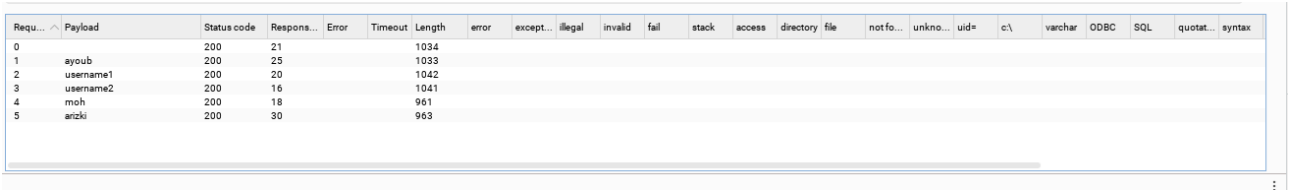
Q2. Les informations affichées par le comparateur sont-elles exploitables pour tenter une énumération ?

➓ Vérification des informations dans le comparateur :

-Après avoir testé plusieurs logins avec l'outil Intruder de BurpSuite, les réponses ont été comparées à l’aide du Comparer.
-Contrairement au mode non sécurisé, en mode sécurisé, les réponses sont moins explicites. L’application ne renvoie rien n indiquant pas si le login existe.

➓ Résultat de l'attaque en mode sécurisé :

- Les informations obtenues dans le comparateur ne permettent plus de distinguer facilement les logins valides des non valides. L'application sécurisée renvoie un message identique pour les deux cas, empêchant ainsi l'énumération des logins à l’aide des différences dans les réponses du serveur.



Q3) 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.

-Le code implémente un service web SOAP sécurisé pour la gestion des comptes utilisateurs, il comprend des fonctionnalités telles que la création, la mise à jour, la suppression et la récupération d'informations utilisateurs. Le niveau de sécurité, défini dans la session (de 0 à 5), détermine si les données sont encodées avant d'être renvoyées dans les réponses SOAP. En mode non sécurisé (niveaux 0 et 1), le code est vulnérable aux attaques comme les injections SQL car les données ne sont pas encodées, laissant potentiellement des failles exploitables. En revanche, aux niveaux de sécurité 2 à 5, la méthode encodeForHTML() est activée pour encoder les données sensibles telles que les noms d'utilisateur et les signatures avant leur affichage ou envoi dans les réponses. Cette fonction transforme les caractères spéciaux (comme <, >, et &) en équivalents HTML sécurisés, empêchant ainsi l'injection de scripts malveillants dans les pages web. Par exemple, un utilisateur malveillant qui tenterait d'injecter un script via un champ de formulaire verrait ce script converti en texte inoffensif, empêchant son exécution. Chaque méthode du service (getUser, createUser, updateUser, deleteUser) suit une logique sécurisée : les paramètres sont validés avec la fonction assertParameter(), les erreurs sont traitées de manière générique pour ne pas révéler d'informations sensibles aux attaquants (comme l'existence d'un utilisateur), et les sorties sont encodées pour éviter les injections. De plus, les messages d'erreur restent toujours génériques (par exemple, « User does not exist » ou « Inserted account »), évitant les fuites d'informations qui pourraient permettre des attaques d'énumération de logins.

```
   /* Example SQL injection: jeremy' union select username,password from accounts -- */

f (session_status() == PHP_SESSION_NONE){
   session_start();
// end if

f (!isset($_SESSION["security-level"])){
   $_SESSION["security-level"] = 0;
// end if

   /* ----------------------------------------
    * Constants used in application
   * ---------------------------------------- */
   require_once('../../includes/constants.php');
   require_once('../../includes/minimum-class-definitions.php');

   try{
           switch ($_SESSION["security-level"]){
                   case "0": // This code is insecure
                   case "1": // This code is insecure
                           $lEncodeOutput = FALSE;
                           break;

                   case "2":
                   case "3":
                   case "4":
                   case "5": // This code is fairly secure
                           $lEncodeOutput = TRUE;
                           break;
           }//end switch

   } catch (Exception $e) {
           echo $CustomErrorHandler->FormatError($e, "ws-user-account.php: Unable to parse session");
   }// end try:
```