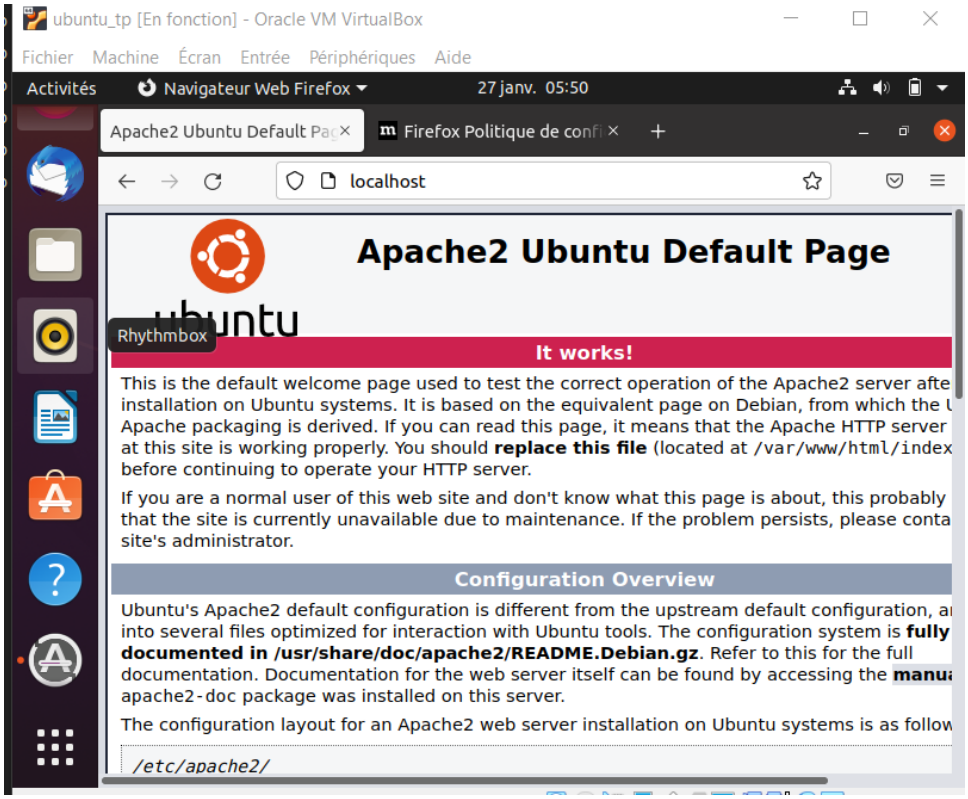
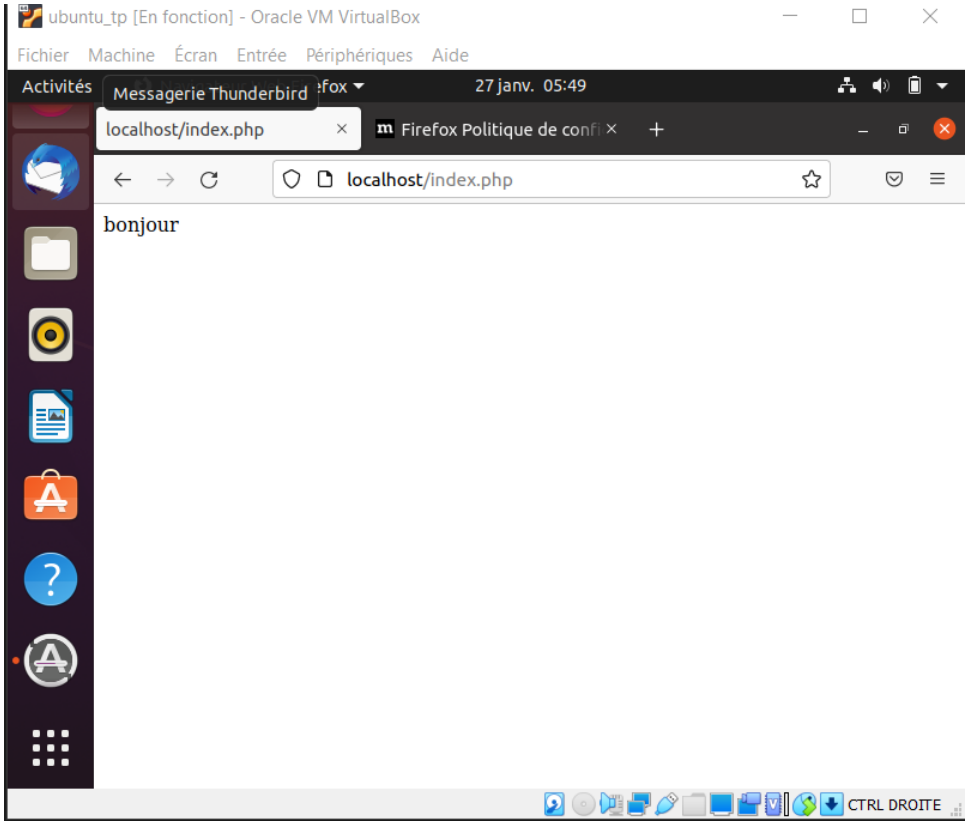
Jonathan Bikambidi

Tp_Owasp

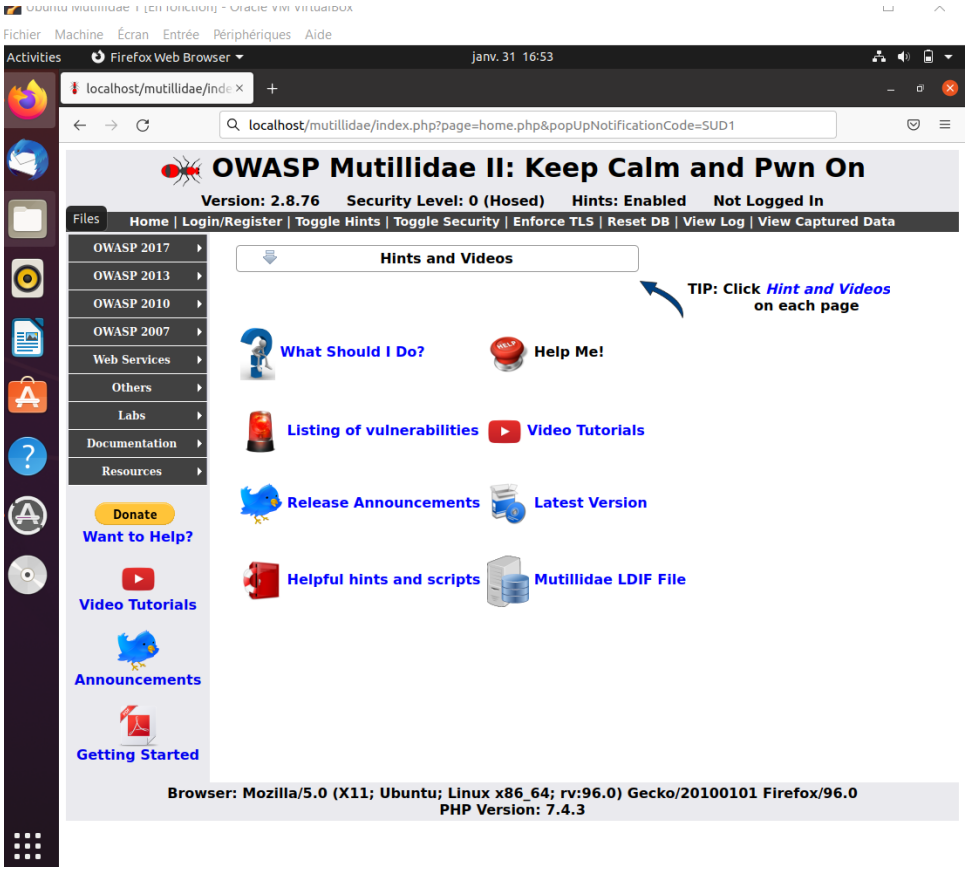
Installation apche2



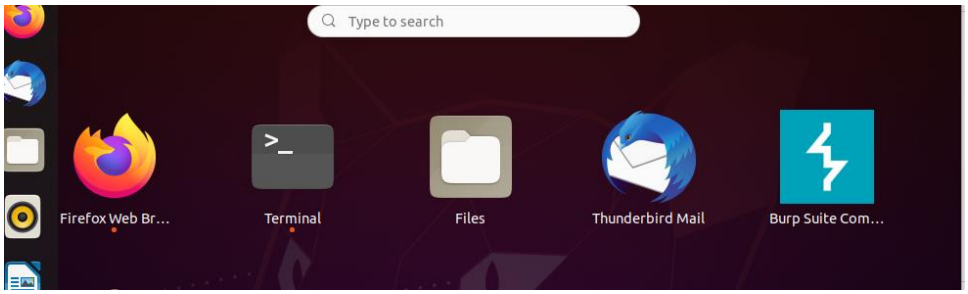
Installation Php

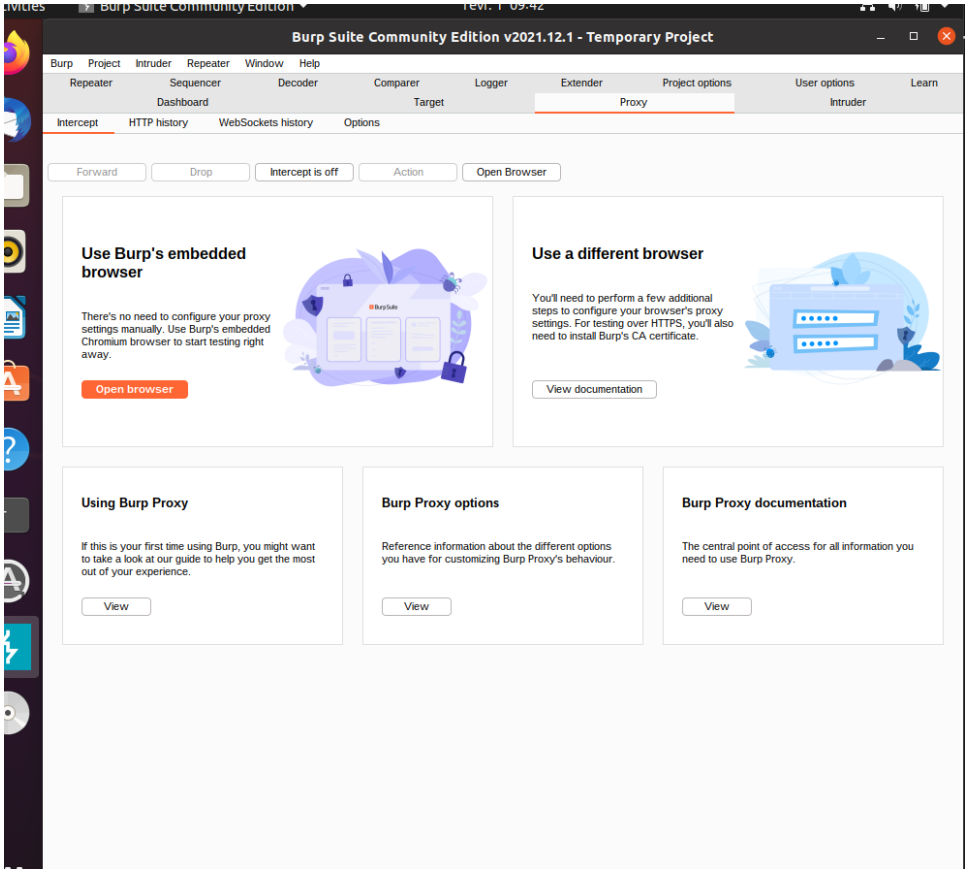


Nouvelle version mutillidae II

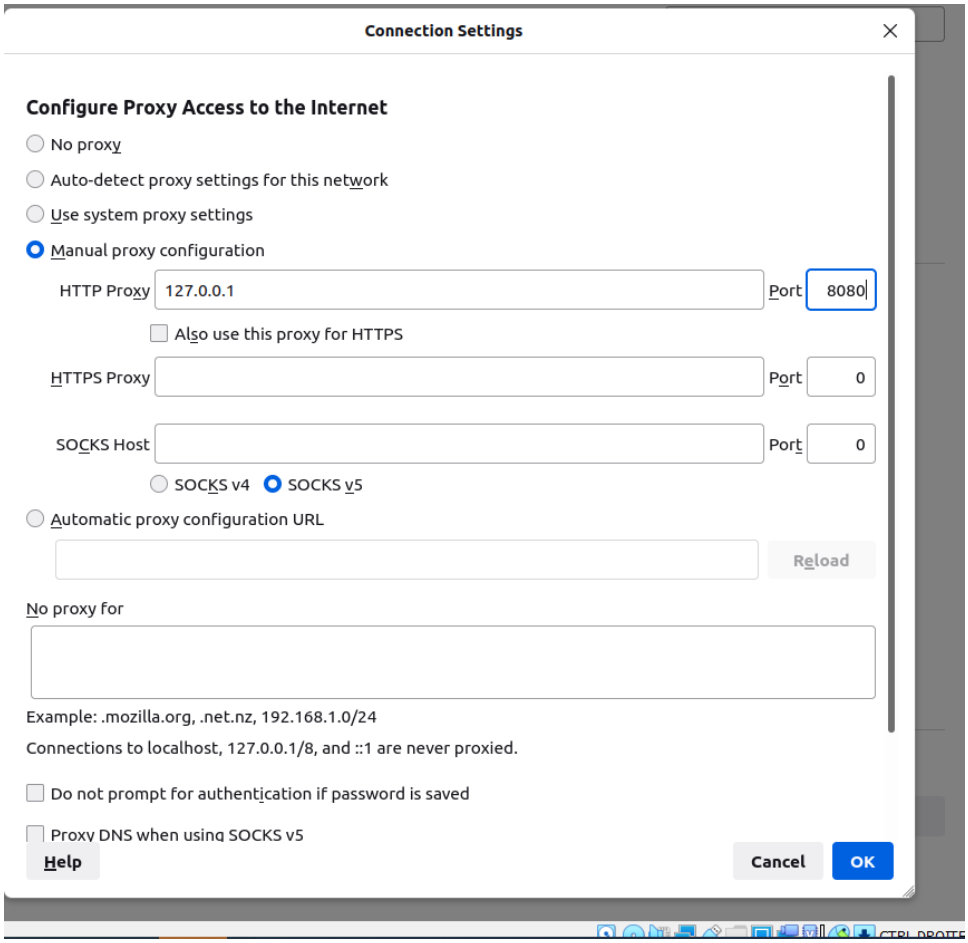


J’ai dû installer burp Suite dans ma machine Ubuntu





Je configure le proxy en rajoutant l’adresse suivante : 127.0.0.1 y compris le port : 8080



J'ai créé un compte et j'ai ensuite saisi l'identifiant et le mot de passe dans les champs, au moment de cliquer sur submit, j'intercepte les données sur Burp Suite

Lorsque je rentre une simple quotte dans le champ password, une erreur SQL apparait, l'application est une vulnérable à l'injection SQL.

Error: Failure is always an option and this situation proves it	
Line	49
Code	0
File	/var/www/mutillidae/process-login-attempt.php
Message	Error executing query: You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near '''''' at line 1
Trace	#0 /var/www/mutillidae/index.php(96): include() #1 {main}
Diagnotic nformation	SELECT * FROM accounts WHERE username='jonathan' AND password=''''

Premier Défi : extraction des données de tous les utilisateurs

En rentrant une simple quotte dans le champ password, une erreur apparait avec cette requête SQL :

Je peux modifier cette requête en ajoutant dans le champ password « ' OR '0' = '0 » pour rendre la condition toujours vraie et tout afficher depuis la table « accounts »

Ce qui donne la requête suivante : SELECT * FROM accounts WHERE username=’jonathan’ AND password=’ ' OR '0' = '0 ’

J’obtiens la liste de tous les utilisateurs avec leurs informations :

Username=admin
Password=adminpass
Signature=g0t r00t?

Username=adrian
Password=somepassword
Signature=Zombie Films Rock!

Username=john
Password=monkey
Signature=I like the smell of confunk

Username=jeremy
Password=password
Signature=d1373 1337 speak

Username=bryce
Password=password
Signature=I Love SANS

Username=samurai
Password=samurai
Signature=Carving fools

Username=jim
Password=password
Signature=Rome is burning

Username=bobby
Password=password
Signature=Hank is my dad

Username=simba
Password=password
Signature=I am a super-cat

Deuxième défi : Passer outre une authentification

J’essaie de me connecter en tant qu’Admin

Q2) Pour se protéger de cette attaque, le niveau de sécurité 1 (Client Side Security) est nécessaire. Il contrôle l’input value et vérifie s’il n’y a pas de caractère dangereux et typique de la faille SQLI comme les cotes. Il renvoie un message d’alerte.

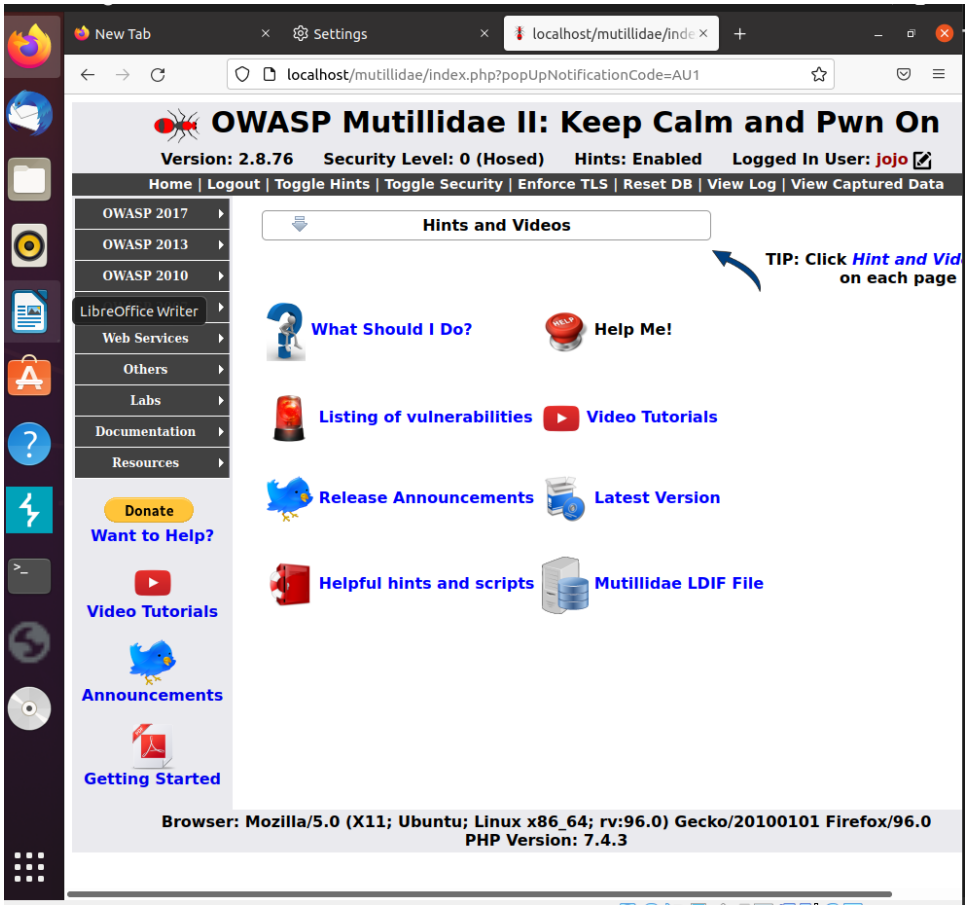
Q3) A ce niveau, les contrôles effectués sont des contrôles sur la saisie de l’utilisateurs. Ce sont des contrôles sur les caractères entrés par l’utilisateurs et empêche la saisi de quotte ou autres caractères spéciaux., en en JavaScript, concernant le HTML, seule la taille des chaînes de caractère est contrôlée.

Q4) Non, le contrôle HTML ne suffit pas à se protéger de cette attaque car il est facilement contournable et parce que le traitement des informations se fait coté PHP.

Q5) La validation JavaScript est enclenchée au clic du bouton, le formulaire est alors envoyé et la validation est enclenché.

```
var lValidateInput = "TRUE"

function onSubmitOfForm(/*HTMLFormElement*/ theForm){
  try{
    var lUnsafeCharacters = /[`~!@#$%^&*()-_=+\[\]{}\\|;':",./<>?]/;

    if(lValidateInput == "TRUE"){
      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 at
        tempts.\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 onSubmitOfForm(/*HTMLFormElement*/ theForm)
```

Je suis bien connecté en tant que jojo



Je place donc mon BurpSuite en intercept is on puis je clique sur ''Home''. J'ai bien intercepté la requête :

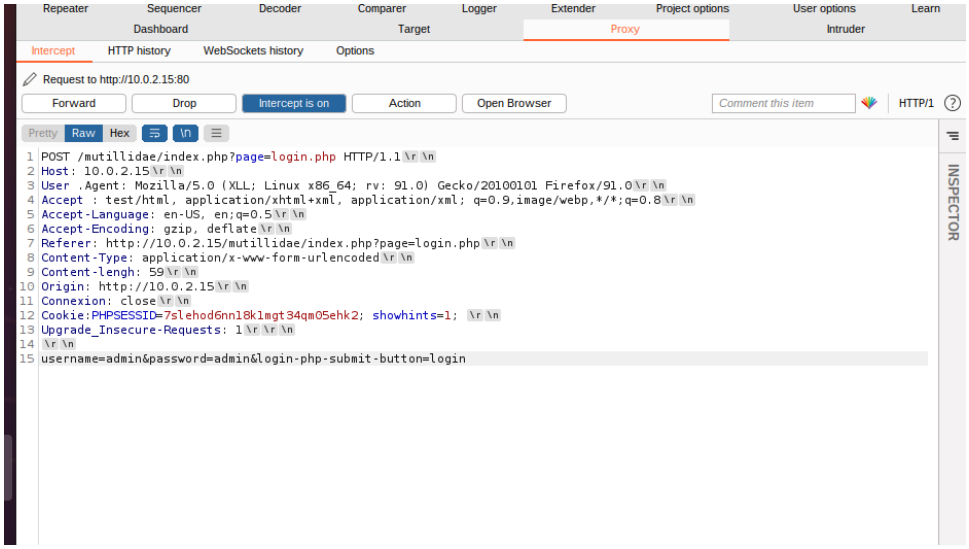
Q2) Ici on peut me voir en présence des informations permettant d'identifier ma session actuelle.

On peut voir le PHP SESS ID ainsi que le cookie ''ujdd'', je peux désormais remplacer mon id par le ''1'' par exemple qui correspondrait naturellement au premier utilisateur crée dans la base de données, potentiellement le compte de l'administrateur.
En effectuant cette manipulation, je me retrouve donc bien connecté en administrateurs.

Deuxième défi : Force brute sur mot de passe

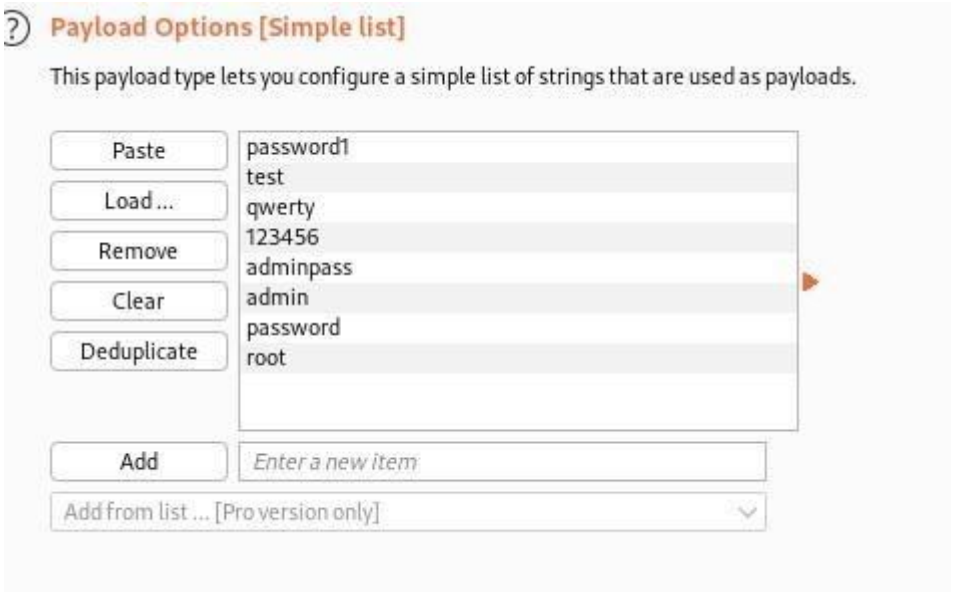
Travail à faire 3

Positionner sur la page de login avec le champ l'identifiant ''admin'' entré, le proxy activé et Burp Suite positionné sur intercept is on :

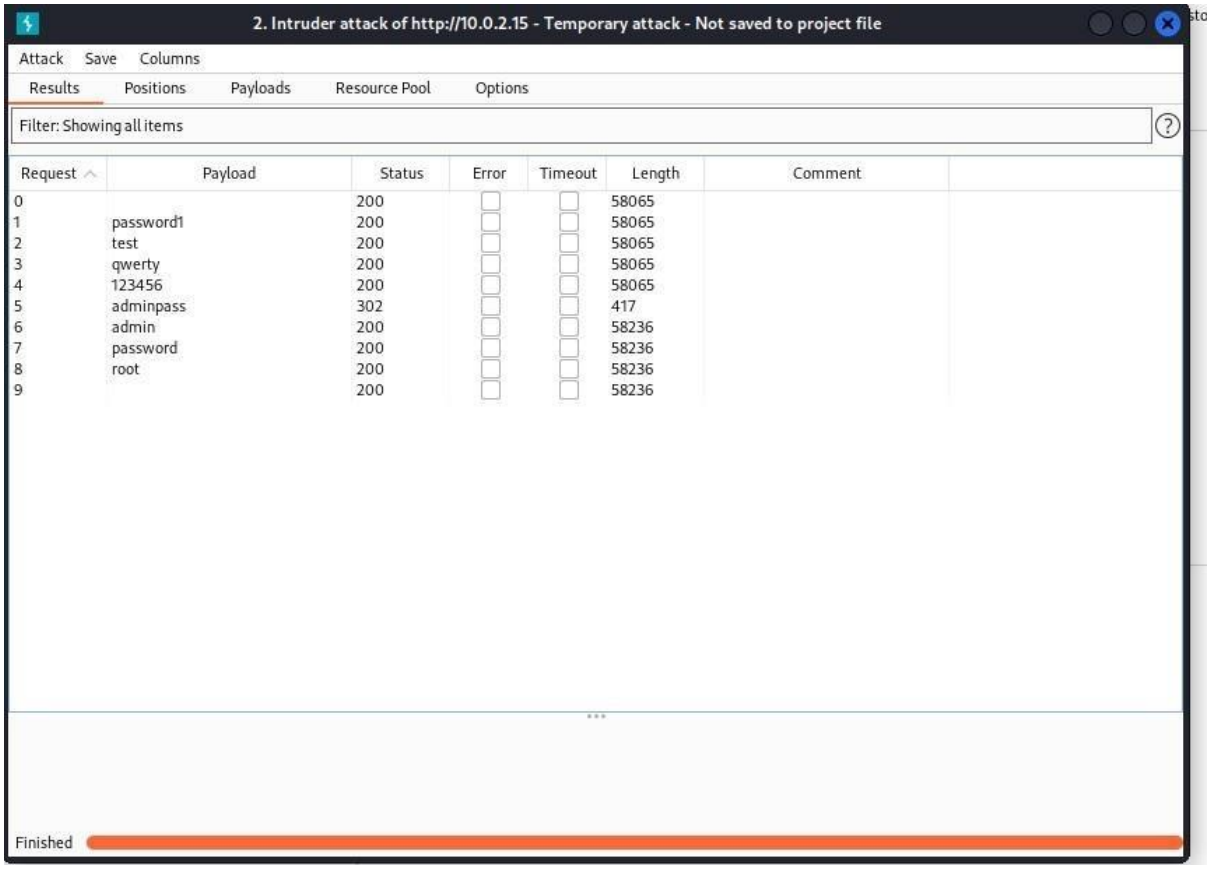


Q2) Lancement de l'attaque.

La requête est ensuite envoyée dans l'onglet intruder. Je conserve que ''admin '' comme variable. Cela va nous servir à lancer l'attaque brut force



Le dictionnaire de mot de passe est bien chargé. Je lance l'attaque :


2. Intruder attack of http://10.0.2.15 - Temporary attack - Not saved to project file
Attack
Save
Columns
Results
Positions
Payloads
Resource Pool
Options
Filter: Showing all items
Request
Payload
Status
Error
Timeout
Length
Comment
0 200 58065
1 password1 200 58065
2 test 200 58065
3 qwerty 200 58065
4 123456 200 58065
5 adminpass 302 417
6 admin 200 58236
7 password 200 58236
8 root 200 58236
9 200 58236
Finished