



Force brute sur un mot de passe

Sommaire

Définition.....	3
Force brute sur un mot de passe.....	4
Codage sécurisé et analyse du code source.....	7

Définition

L'attaque brute force consiste à connaître le login et le mot de passe de la victime en essayant de se connecter à plusieurs reprises.

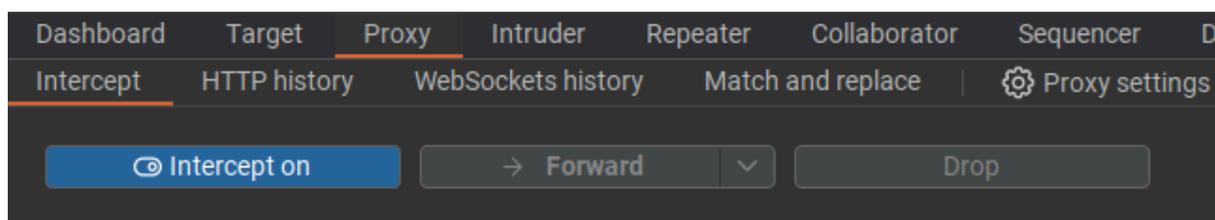
Force brute sur un mot de passe

L'utilisateur **admin** va être la cible pour permettre de se connecter à ce compte. Avec l'attaque brute force, plusieurs combinaisons de mots de passe vont être utilisés séquentiellement.

Sur le site web "mutillidae", le niveau de sécurité est défini à 0, ce qui permet d'effectuer des attaques.



Côté proxy, l'interception est activée pour permettre de récupérer des informations sur les requêtes ainsi que des réponses.

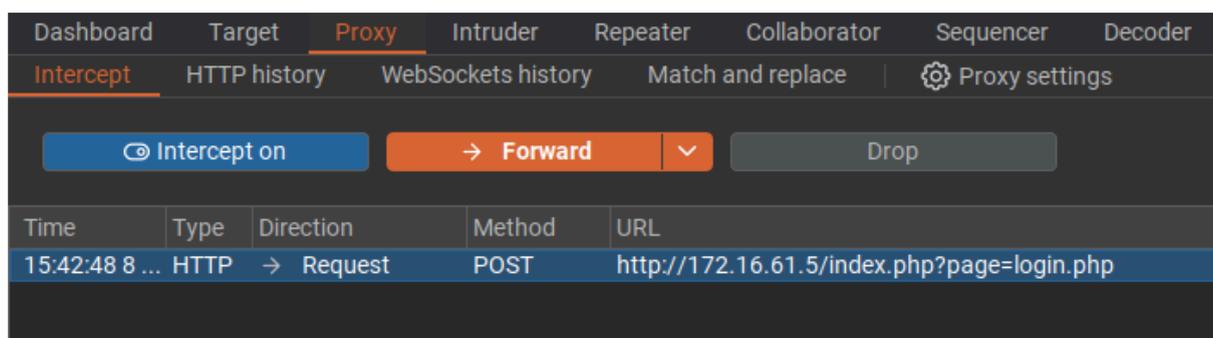


Dans le formulaire, le nom d'utilisateur est **admin** et le mot de passe erroné a été saisi pour intercepter la requête dans BurpSuite.

Please sign-in

Username

Password



```

Request
Pretty Raw Hex
1 POST /index.php?page=login.php HTTP/1.1
2 Host: 172.16.61.5
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.16.61.5/index.php?page=login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 63
10 Origin: http://172.16.61.5
11 Connection: keep-alive
12 Cookie: PHPSESSID=7tud3bj99914v51gu98age61sq; showhints=0
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 username=admin&password=dsfsdfsfs&login-php-submit-button=Login

```

Résultat, au niveau de la requête, on peut voir à la dernière ligne qu'il y a deux paramètres : username (comme valeur "admin") et password avec la valeur un mot de passe erroné. Avec cette requête, on peut tester toutes les combinaisons des mots de passe : c'est l'intruder qui se charge de ça. Pour cela, cliquer sur "Send to Intruder" à partir du menu.

Time	Type	Direction	Method	URL
15:42:48 8 ...	HTTP	→ Request	POST	http://172.16.61.5/index.php?page=login.php
			Scan	
			Send to Intruder	Ctrl+I

Dans l'intruder, on met une liste de mots de passe pour permettre de se connecter au compte **admin**. Côté requête, il faut remplacer la valeur de la clé "password" par cette liste de mots de passe qui vont être testées une par une. Pour cela, sélectionner le mot de passe erroné et cliquer sur le bouton "Add §"

Payload configuration

This payload type lets you configure a simple list of strings that are used as payloads.

Paste

Load...

Remove

Clear

Deduplicate

Add

Add from list... [Pro version only]

- jesaispas
- coucou
- wikipedia
- linux
- windows
- adminpass
- Enter a new item

Add § Clear § Auto §

```

1 POST /index.php?page=login.php HTTP/1.1
2 Host: 172.16.61.5
3 User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:121.0) Gecko/20100101 Firefox/121.0
4 Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/png,image/svg+xml,*/*;q=0.8
5 Accept-Language: fr,fr-FR;q=0.8,en-US;q=0.5,en;q=0.3
6 Accept-Encoding: gzip, deflate, br
7 Referer: http://172.16.61.5/index.php?page=login.php
8 Content-Type: application/x-www-form-urlencoded
9 Content-Length: 63
10 Origin: http://172.16.61.5
11 Connection: keep-alive
12 Cookie: PHPSESSID=7tud3bj99914v51gu98age61sq; showhints=0
13 Upgrade-Insecure-Requests: 1
14 Priority: u=0, i
15
16 username=admin&password=@dsfsdfsfs&login-php-submit-button=Login

```

Pour lancer l'attaque brute force, cliquer sur le bouton "Start Attack".

Request	Payload	Status code	Response...	Error	Timeout	Length	Comment
0		200	124			55511	
1		200	21			55510	
2	jesaispas	200	13			55510	
3	coucou	200	29			55511	
4	wikipedia	200	15			55510	
5	linux	200	33			55511	
6	windows	200	13			55510	
7	adminpass	302	19			480	

Dans le tableau présenté ci-dessus, on constate que tous les payloads (qui sont en réalité des mots de passe) obtiennent un statut de type 200, pour dire que la réponse a bien été reçue.

Par contre, dans la dernière ligne du tableau, on remarque que le statut n'est pas le même que celui des autres : elle est de type 302, désignant une redirection. Donc cela signifie que le mot de passe nommé "adminpass" correspond bien à celui du compte admin.

Codage sécurisé et analyse du code source

Après avoir réalisé ce type d'attaque, côté du site "mutillidae", le niveau de sécurité est défini à 5. Les procédures sont les mêmes pour réaliser l'attaque brute force.



The screenshot shows the OWASP Mutillidae II application header with the following information: Version: 2.11.15, Security Level: 5 (Secure), Hints: Disabled, and Not Logged In. Below the header is a table titled "Intruder attack results filter: Showing all items" with the following data:

Request	Payload	Status code	Response received	Error	Timeout	Length	Comment
0		200	49			55239	
1	admin	200	15			55238	
2	coucou	200	34			55239	
3	bonjour	200	19			55238	
4	mini	200	47			55239	
5	maxi	200	13			55238	
6	utilisateur	200	36			55239	
7	adminpass	302	9			505	
8	simple	200	30			55410	
9	basique	200	11			55409	

Même si le niveau de sécurité est élevé à la fois côté client et côté serveur, ça ne suffit pas pour éviter ce type d'attaque car dans la ligne du tableau où il y a "adminpass", on obtient toujours le statut de redirection HTTP, signifiant que l'attaquant a réussi à se connecter en tant qu'administrateur.

```
case "5": // This code is fairly secure
/*
 * Concerning SQL Injection, use parameterized stored procedures. Parameterized
 * queries is not good enough. You cannot use least privilege with queries.
 */
$!EnableJavaScriptValidation = TRUE;
$!EnableHTMLControls = TRUE;
$!ProtectAgainstMethodTampering = TRUE;
$!EncodeOutput = TRUE;
break;
```

Grâce à la variable `$!EnableHTMLControls`, on peut ajouter des attributs dans la balise `<input>` pour ajouter des vérifications côté client, dont notamment le nombre de caractères minimales dans le champ du mot de passe.

```
<input type="password" name="password" size="15"
  <?php
    if ($!EnableHTMLControls) {
      echo('minlength="1" maxlength="15" required="required"');
    } // end if
  ?>
/>
```

Dans la balise `<input>` de type `password`, on voit qu'on peut mettre au minimum 1 caractère et au maximum 15 caractères.

```

<?php
    if($lEnableJavaScriptValidation){
        echo "var lValidateInput = \"TRUE\"" . PHP_EOL;
    }else{
        echo "var lValidateInput = \"FALSE\"" . PHP_EOL;
    }// end if
?>

function onSubmitOfForm(/*HTMLFormElement*/ theForm){
    try{
        if(lValidateInput == "TRUE"){
            var lUnsafeCharacters = /[\\W]/g;
            if (theForm.username.value.length > 15 ||
                theForm.password.value.length > 15){
                alert('Username too long. We c

                return false;
            }// end if
        }
    }
}

```

Puis, côté JavaScript, on vérifie si le nom d'utilisateur ou le mot de passe ne dépasse pas 15 caractères. Sinon, il affichera une alerte.

Même avec le niveau de sécurité définie à 5, ça ne suffit pas pour forcer les utilisateurs à sécuriser leurs mots de passe.

Pour faire face à cette faille de sécurité, il est possible :

- de faire des vérifications côté serveur car cela permet à l'utilisateur de ne pas manipuler les validations à la fois dans le HTML et dans le JavaScript et cela ajoute une couche de sécurité supplémentaire à l'application web.
- de limiter le nombre de tentatives de connexion à un compte. Par exemple, si on a raté 5 fois, alors l'application va mettre en place un délai de plusieurs minutes voire plusieurs heures pour atténuer l'attaque brute force.