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

3

Wildcard scoped access token stealing.

State	Resolved (Closed)
Disclosed	**January 9, 2019 2:02am +0530**
Reported To	Linode
Asset	https://www.linode.com (Domain)
Weakness	Improper Access Control - Generic
Bounty	$2,000
Severity	Critical (9.6)
Participants	
Visibility	Disclosed (Full)

Collapse

bugdiscloseguys submitted a report to **Linode**. Sep 30th (4 months ago)

Hey team,

Summary :

By chaining multiple vulnerabilities it is possible to steal access token of users with `*` scope.

Description

1. Setting & accessing cookies for `.linode.com/*`

Because Linode provides a feature to resolve a Linode instance to `<tenant>.members.linode.com` it is possible to set a cookie for `.linode.com` on any path. We can also access any cookie which is set to `.linode.com` as Domain.

2. Linode community sets `sessionid` cookie to domain `.linode.com`

Linode community session is set to `.linode.com` which allows an attacker to steal it, The community portal also interact with API for notification resource. For which a token is also gets saved in source of page but the token is limited to very small scope i.e `events:modify`.

3. Linode community client app have access to `*` (wildcard) scope. (Not a issue)

As the title suggests, the client app can access * scope but if only given in the scopes while oauth authorization.

Understanding the login flow of Linode community portal.

On clicking `Login` A GET request is made to `https://www.linode.com/community/questions/login?next=/community/` following is the response of the request.

```
HTTP/1.1 302 Found
Server: nginx
Date: Sat, 29 Sep 2018 23:36:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
Vary: Cookie
Location: https://login.linode.com/oauth/authorize?scopes=events%3Amodify&state=bce45f7c-6a37-46c7-9ede-c9979c152081
```

```
Set-Cookie: sessionid=dgagljcsrcg0m3klfd2o16x9q1smbgvd; Domain=.linode.com; expires=Sat, 13-Oct-2018 23:36:51 GMT; H
.....
```

Which does following

- Sets cookie :

  `sessionid=dgagljcsrcg0m3klfd2o16x9q1smbgvd` ; which is linked to state token `state=bce45f7c-6a37-46c7-9ede-c9979c152081` .

- Redirect to OAuth page :
  `https://login.linode.com/oauth/authorize?scopes=events%3Amodify&state=bce45f7c-6a37-46c7-9ede-c9979c152081&client_id=a38f156de7fa9819c110&redirect_uri=https%3A%2F%2Fwww.linode.com%2Fcommunity%2F&response_type=code`

- If logged in on `https://login.linode.com/` User get redirected to

`https://www.linode.com/community/?state=bce45f7c-6a37-46c7-9ede-c9979c152081&code=6f422a104f5bf039f9dc`

The state parameter token is cross checked against the earlier seted sessionid, If verification succeed, We get this response;

```
TTP/1.1 302 Found
Server: nginx
Date: Sat, 29 Sep 2018 23:04:02 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
Vary: Cookie
Location: /community/
Set-Cookie: sessionid=qaff7xdtoxxhnc6tds7ym2d7d9lpweci; Domain=.linode.com; expires=Sat, 13-Oct-2018 23:04:02 GMT; H
....
```

Which does following

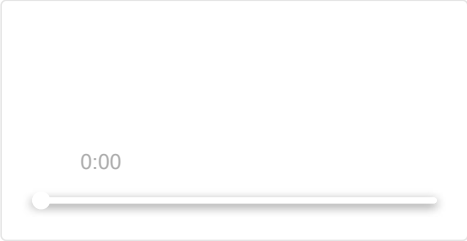
- sessionid gets reset which is an actual session for community portal and user get logged in into community portal.

Exploiting

Our first goal is that we want `*` scoped access token from community portal. if we want user to login into community portal with `*` scope access token we need to bypass the CSRF mechanism. the login mechanism looks for `sessionid` and cross verify it against the given `state` token.
because we can set `.linode.com` we can bypass this :)

Once we we bypass login CSRF, We make user login to community with `*` scope and because the community portal session will be set to `.linode.com` we can fetch the legit session and finally extract the token from source. :)

PoC



https://li859-243.members.linode.com/exploit.php this exploit expects you to be logged in at `login.linode.com` so make sure of that.

Impact

Access token stealing.

5 attachments:
F353168: Screenshot_143.png
F353177: 2018-09-30_06-46-44.mp4
F353178: exploit.php
F353179: setcookie.php
F353180: stc.php

bugdiscloseguys updated the severity to High. Sep 30th (4 months ago)

bugdiscloseguys posted a comment. Sep 30th (4 months ago)
We can then use that token to login at cloud.linode.com.
`https://cloud.linode.com/oauth/callback#access_token={STOLEN_TOKEN_HERE}&token_type=bearer&expires_in=7200&scope=*&state=ce06904a-a073-4856-b6aa-78ee570b5476&return=https://cloud.linode.com/oauth/callback?returnTo=/`

linode_ctarquini posted a comment. Oct 1st (4 months ago)
Yikes. Thanks for your report, we're going to look into mitigating this asap. We should have a preliminary bounty sent out by tomorrow afternoon.

bugdiscloseguys posted a comment. Oct 1st (4 months ago)
@linode_ctarquini
Thanks for your reply, I have a suggestion regarding the fix.

The fix should move <tenant>.members.linode.com to some other domain, As long as the attacker have access on <tenant>.members.linode.com, Attacker can abuse this in other bugs too.

I have a self stored XSS on manager.linode.com but because of login CSRF protection at manager.linode.com i was not able to make user login to my account earlier however i am able to now by setting cookies for .linode.com (SESSIONID) which will make victim login into attacker account at manager.linode.com, The XSS will execute in attacker session on victim browser but my friend and i have a theory on stealing victim's linode account details using it and we're working on it.

My point of adding this is that you should move the tenant thing to some sandbox domain.

Regards,
Harsh

linode_ctarquini changed the status to ○ **Triaged**. Oct 1st (4 months ago)

linode_ctarquini posted a comment. Oct 2nd (4 months ago)
Hey @bugdiscloseguys,

We do have an open item to move everyone off of *.linode.com (Linode rDNS and Nodebalancers) due to security concerns. Unfortunately, this is a non-trivial to change without breaking user setups so in the short term we're looking to do the following:

- Reduce the scope of sessionid
- Ensure that new session ids are always generated upon login on any *.members.linode.com services (mitigate session fixation attackers where the attacker sets the session id and the user logs in later)
- Begin discussion to sunset *.members.linode.com

> The XSS will execute in attacker session on victim browser but my friend and i have a theory on stealing victim's linode account details using it and we're working on it.

Looking forward to hearing more about this! Let me know if you have any questions for us.

linode_ctarquini updated the severity from High to Critical (9.6). Oct 3rd (4 months ago)

Linode rewarded bugdiscloseguys with a **$2,000** bounty. Oct 3rd (4 months ago)

We've rated this issue's severity as critical and have issued an initial bounty of $2,000. We will update you once mitigations are deployed or if additional bounties are issued.

bugdiscloseguys posted a comment. Oct 3rd (4 months ago)
Woah! thanks :) Always good to work with you guys :)

linode_ctarquini posted a comment. Oct 3rd (4 months ago)
Hey @bugdiscloseguys,

> Woah! thanks :) Always good to work with you guys :)

The feelings mutual! Thank you for your very well-written/detailed reports

We deployed a quick mitigation to reduce the scope of the sessionid token and this seems to have prevented the PoC from working. It looks like you can set a cookie with higher precedence via *.members.linode.com (force a user to login as an attacker) still so I'm not 100% convinced this isn't exploitable yet.

If you find a way around this mitigation, please let me know and I'll send another bounty your way. We're going to keep investigating this on our end as well.

bugdiscloseguys posted a comment. Oct 3rd (4 months ago)
This looks fixed, Cookies are now set to www.linode.com which won't be accessible by *.members.linode.com. Still finding way to exploit the XSS.

linode_ctarquini closed the report and changed the status to ○ **Resolved**. Oct 4th (4 months ago)
Since this looks to be mitigated, I'm going to close this report. We look forward to seeing more findings from you soon!

bugdiscloseguys posted a comment. Updated Oct 4th (4 months ago)
@linode_ctarquini I'm looking to do a short write-up cause i think this* was a good one, should i include Linode or completely remove Linode details? I'm fine with whatever best for you/Linode.

linode_ctarquini posted a comment. Oct 9th (4 months ago)
Hey @bugdiscloseguys,

Are you publishing via the Hacktivity feed? This would let us redact any sensitive information before public disclosure

bugdiscloseguys posted a comment. Oct 13th (4 months ago)
Sorry for late reply, sick from last couple of days.

Yes we can make use of hacktivity publish feature (https://docs.hackerone.com/hackers/publishing-external-vulnerabilities.html) or i can write a short blog. Whatever best for you, I will replace all occurances of `linode` with `redcated` in both type of disclosures.

bugdiscloseguys posted a comment. Oct 13th (4 months ago)
Here's the modified report which i will ask HackerOne to publish. Have a look, Let me know if you want to redact any more information.

Linode -> Redacted
Community -> Forum
Cloud -> app
scopes=events:modify -> scopes=events

Summary :

`login.redacted.com` - OAuth provider
`<tenant>.members.redacted.com` - Attacker controlled asset
`www.redacted.com/forum` - Redacted's forum

By chaining multiple vulnerabilities it is possible to steal access token of users with `*` scope.

Description

1. Setting & accessing cookies for `.redacted.com/*`

Redacted provides a feature to resolve <your-tenant>.members.redacted.com to your server. This allows us to set a cookie for `.redacted.com` on any path. We can also access any (including HTTPonly) cookie which is set to/for `.redacted.com`.

2. Redacted's forum sets `sessionid` cookie to domain `.redacted.com`

Redacted's forum session is set to `.redacted.com` which allows an attacker to steal it, The forum portal also interact with API for notification resource. For which a token is also gets saved in source of page but the token is limited to very small scope i.e `events`.

3. Redacted's forum client app have access to `*` (wildcard) scope. (minor issue)

As the title suggests, the client app can access `*` scope but if only given in the scopes while oauth authorization.

Understanding the login flow of Redacted's forum.

On clicking `Login` A GET request is made to `https://www.redacted.com/forum/questions/login?next=/forum/` following is the response of the request.

```
HTTP/1.1 302 Found
Server: nginx
Date: Sat, 29 Sep 2018 23:36:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
Vary: Cookie
Location: https://login.redacted.com/oauth/authorize?scopes=events&state=bce45f7c-6a37-46c7-9ede-c9979c152081&client
Set-Cookie: sessionid=dgagljcsrcg0m3klfd2o16x9q1smbgvd; Domain=.redacted.com; expires=Sat, 13-Oct-2018 23:36:51 GMT;
.....
```

Which does following

- Sets cookie :

  `sessionid=dgagljcsrcg0m3klfd2o16x9q1smbgvd` ; which is linked to state token `state=bce45f7c-6a37-46c7-9ede-c9979c152081`.

- Redirect to OAuth page : `https://login.redacted.com/oauth/authorize?scopes=events&state=bce45f7c-6a37-46c7-9ede-c9979c152081&client_id=a38f156de7fa9819c110&redirect_uri=https%3A%2F%2Fwww.redacted.com%2Fforum%2F&response_type=code`

- If logged in on `https://login.redacted.com/` User get redirected to

`https://www.redacted.com/forum/?state=bce45f7c-6a37-46c7-9ede-c9979c152081&code=6f422a104f5bf039f9dc`

The state parameter token is cross checked against the earlier seted sessionid, If verification succeed, We get this response;

```
TTP/1.1 302 Found
Server: nginx
Date: Sat, 29 Sep 2018 23:04:02 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 0
Connection: close
Vary: Cookie
Location: /forum/
Set-Cookie: sessionid=qaff7xdtoxxhnc6tds7ym2d7d9lpweci; Domain=.redacted.com; expires=Sat, 13-Oct-2018 23:04:02 GMT;
....
```

Which does following

- sessionid gets reset which is an actual session for forum portal and user get logged in into forum portal.

Exploiting

Our first goal is that we want `*` scoped access token from forum. if we want user to login into forum portal with `*` scope access token we need to bypass the CSRF mechanism. the login mechanism looks for `sessionid` and cross verify it against the given `state` token. because we can set `.redacted.com` we can bypass this :)

Once we we bypass login CSRF, We make user login to forum with `*` scope and because the forum portal session will be set to `.redacted.com` we can fetch the legit session and finally extract the token from source. :)

PoC

https://attacker-tenant.members.redacted.com/exploit.php this exploit expects you to be logged in at `login.redacted.com` so make sure of that.

Impact

Access token stealing.

Login at
https://app.redacted.com/oauth/callback#access_token=%7BSTOLEN_TOKEN_HERE%7D&token_type=bearer&expires_in=7200&scope=*&return=https://app.redacted.com/oauth/callback?returnTo=/

bugdiscloseguys requested to disclose this report. Jan 8th (22 days ago)
Would be great If you don't have any problem disclosing this too.

linode_bdorsey posted a comment. Jan 8th (21 days ago)
Hey @bugdiscloseguys, apologies for dropping the ball with regards to your request to disclose the redacted version of this report to public HackerOne. I will be re-approaching this topic of discussion with the security team this week and will have an answer for you this week.

linode_bdorsey agreed to disclose this report. Jan 9th (21 days ago)
Is public disclosure of the redacted version you've written above something you would still be interested in? If so, I can try and advocate for this with the rest of the team. In the interim, I've accepted your request to disclose this report to all members of our private program.

This report has been disclosed. Jan 9th (21 days ago)

bugdiscloseguys posted a comment. Jan 9th (21 days ago)
Yeah i would like to do a public disclosure. But ofc. not necessary if that gets you alot of trouble. We can keep it limited to private program disclosure only.