Slack bot token leakage exposing business critical information

By Detectify

Published: 2016-04-28 · Archived: 2026-04-05 23:33:25 UTC

Developers are leaking access tokens for Slack widely on GitHub, in public repositories, support tickets and public gists. They are extremely easy to find due to their structure. It is clear that the knowledge about what these tokens can be used for with malicious intent is not on top of people's minds…yet. The Detectify team shows the impact, with examples, and explains how this could be prevented.

***Update**: Slack has made a follow-up response: "we've revoked the tokens you reported, notified affected users and team owners directly, and that we'll be doing that proactively going forward".*

Slack has become one of the world's most popular corporate tools for communicating internally. The simplicity of the Slack API and its tokens also inspire people to create services using Slack to automate manual tasks. This aligns well with the ChatOps movement, a conversation-driven development widely credited to GitHub. Using ChatOps enables you to do automation only by writing simple commands in a chat.

The problem is that many developers tend to include Slack tokens – credentials tied to their personal Slack account – directly in the code when building Slack bots. These projects are also shared publicly on GitHub. Now, because the code contains these tokens, the developer is actually giving anyone – that finds the token – access to the developer's company's internal chats and files on Slack. And not only that, there's no easy way to see if someone is eavesdropping on the communication.

> The problem is that many developers tend to include Slack tokens – credentials tied to their personal Slack account – directly in the code when building Slack bots. These projects are also shared publicly on GitHub.

The irony is that a lot of these bots are mostly fun "weekend projects". We saw examples of fit bots, reminding you to stretch throughout the day, quote bots, quoting both Jurassic Park ("Ah ah ah…you didn't say the magic word") and Don Quijote. Yet, these hobby projects, in combination with being shared on GitHub, risk exposing both sensitive company information as well as private messages and files for the user that created the token.

In the worst case scenario, these tokens can leak production database credentials, source code, files with passwords and highly sensitive information. The Detectify Team have already been able to find thousands of tokens by simply searching GitHub; and new tokens are becoming publicly available every day.

How to generate Slack tokens – and where the problem occurs

There are multiple open source projects to easily create Slack bots, such as Hubot. Hubot does a great job of using environment variables for access tokens. One reason behind environment variables is to separate credentials from the code completely, getting the environment that runs the code to provide the data depending on where the code runs.

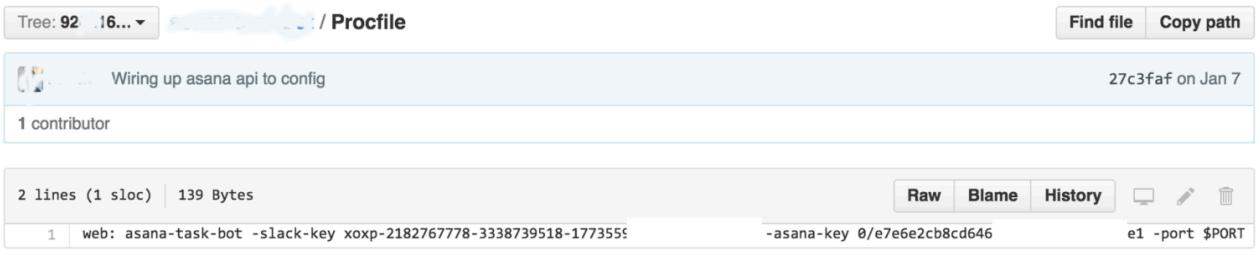
However, developers tend to be creative in how to work with these variables. Here are real life examples where the developers are injecting the tokens inside the code onto the environment, which breaks the whole concept of how environment variables are supposed to be set up:

Executable File | 8 lines (5 sloc) | 180 Bytes

```
1  #!/bin/bash
2
3  export HUBOT_SLACK_TOKEN=xoxb-16568917879-AlcthJ
4  export HUBOT_SLACK_TEAM=          est
5  export HUBOT_SLACK_BOTNAME=localbot
6
7  bin/hubot --adapter slack
```

9 lines (7 sloc) | 400 Bytes

```
1  #!/bin/sh
2
3  heroku config:add HEROKU_URL="http://              .herokuapp.com"
4  heroku config:add HUBOT_GITHUB_ORG="ir          nter"
5  heroku config:add HUBOT_GITHUB_TOKEN="6c                              9ad78678d408"
6  heroku config:add HUBOT_SLACK_BOTNAME="Test-Sl          "
7  heroku config:add HUBOT_SLACK_TEAM="Depl      "
8  heroku config:add HUBOT_SLACK_TOKEN="xoxp-2446364950-2445644913-25778          3a"
```



We also saw examples of where log output from applications was spitting out the Slack tokens in clear text, leaking them when people posted GitHub issues needing help in public projects:

commented 11 days ago

EDIT: I should handle my unwrap first. I will let you know then.

Ok, here's the stacktrace with RUST_BACKTRACE=1 cargo run ` for the above program.

```
r` value: JsonDecode(MissingFieldError("text"))', ../src/libcore/result.rs:741

h71b7c99a21cff148fMt
sure.40876
d8BFx
r::hb8ac08b8a72b9d1bhPs
:h076ba7ed08690e9dnOs

67LFK
9894293632943621 6
da
67
::h995944026564267767

3587514840227558



lacktail/target/debug/slacktail xoxp-3522861600-3523048018-14491818720-          ` (exit code: 101)
```

It looks like slack-rs-api is not handling certain events correctly. What other infor can I provide?

Application Error #120

Closed opened this issue on Nov 19, 2015 · 4 comments

commented on Nov 19, 2015

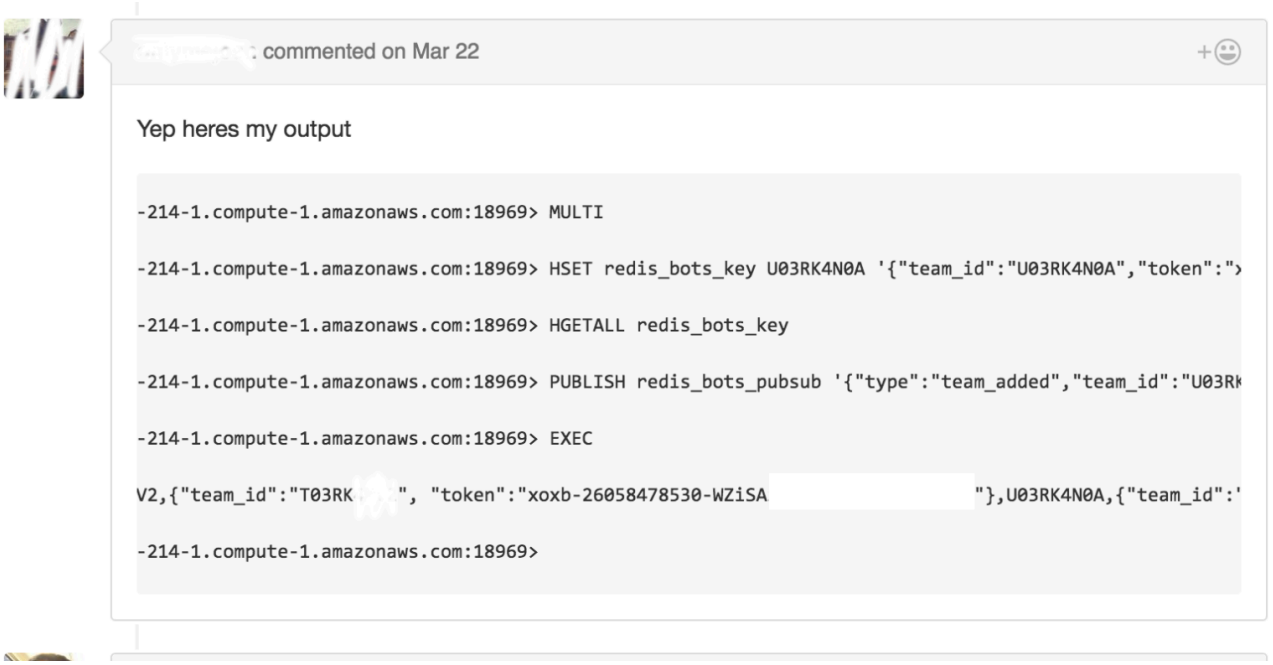
Anyone have any insight on why we would be receiving this application error?

Heroku support responded with the following...

Hi Tom,

I'd recommend taking a look at heroku logs in the CLI, it seems to be connecting to an invalid host:

2015-11-13T01:31:26.462183+00:00 heroku[web.1]: Starting process with command `bin/slackin --channels "$SLACK_CHANNELS" --port 35494 :  .slack.com xoxp-10537258593-10532710582-14401515057`
2015-11-13T01:31:29.389530+00:00 app[web.1]: Fri Nov 13 2015 01:31:29 GMT+0000 (UTC) – fetching
2015-11-13T01:31:29.440592+00:00 app[web.1]: Fri Nov 13 2015 01:31:29 GMT+0000 (UTC) – Error:



How big is the problem?

Independently of Slack, tokens on GitHub are unfortunately very common. But in this case, compared to AWS tokens that were reported being scraped by black hats a few years ago, it's most likely because the developers don't know about the full impact of leaking a Slack token. Also, a bonus is how GitHub's full text index work and the format of Slack tokens, since the tokens use a prefix with a hyphen as a separator:

`xoxp-XXXXXXXXX-XXXXXXXXXXXXXXXXXX`
`xoxb-XXXXXXXXX-XXXXXXXXX-XXXXXXXXX-XXXXXX`

GitHub's search treats hyphenated strings as separate words, and since these prefixes are very distinguishable of being tokens to Slack, you will find a huge amount of Slack tokens on GitHub just by searching for the prefixes.

Using the tokens it's possible to eavesdrop on a company. Outsiders can easily gain access to internal chat conversations, shared files, direct messages and even passwords to other services if these have been shared on Slack. Here are some examples of sensitive messages being shared among people inside Slack teams **(the sensitive information in these examples has been modified or redacted):**

```
"text": "<!here|@here> You can use my email for the XXXXXXX, and `AwezGrowth` as the password.",
"permalink": "https://xxxxxxx.slack.com/archives/dev-growth/p14573712121219",
"user": "U06A19PDF",
"username": "flarsson",
```

```
{"type":"message","user":"U0552AAL9","username":"way","ts":"1450309434.000164","text":"<@U01AUU
A2BA>: what's the vimeo password again?","channel":

"next":{"user":"U01AUUA2BA","username":"jen","ts":"1450309910.000165","type":"message","text":"
banana12"}
```

```
{"title":"","value":"hey there! to connect to our database, here is the connection info. Host:
X.X.X.X Username: fala-ade23 password: csAdd12dsw Database: dms_all_125","short":false}
```


Companies that we've been in contact with to let them know about them leaking their token were surprised of the amount of data that was accessible by this token as per default. One of the most common responses were "I did not know that token gave that much access" which indicates that the process to generate these tokens does not adequately explain how much data could be exposed by abusing the token.

There are different types of tokens in Slack, and some of them has less access than others. The problem is that it is difficult to create a proper limited token whereas creating a very open token with full access is easy.

Who is affected?

Up until now we've identified over 1500 tokens that match the pattern of a Slack token being publicly available on GitHub. These tokens belong to different users and companies; among them Forbes 500 companies, payment providers, multiple internet service providers and health care providers. Renowned advertising agencies that want to show what they are doing internally. University classes at some of the world's best-known schools. Newspapers sharing their bots as part of stories. The list goes on and on.

Our research and results

As we've mentioned, there are different kinds of tokens in Slack. There are also different restrictions to them. To explain this we need to understand what types of API-endpoints there are:

- **Search API.** This API makes it possible to search using a query and get all results back in a list.
- **IM/Groups API.** The private messages sent to a user, both by direct messages and groups.
- **Channels API.** Using these APIs you can gather all data for all public channels in a team. Please understand that public in this sense means that anyone internally in the team can join, the information in this channel is not public per se.
- **Users API.** You will get all users in the complete Slack team together with their e-mail address and user information.
- There are many more APIs, you can find them all here.

Now, these APIs above are essential for someone that wants to eavesdrop on a company. If a token is leaked, depending on the token there are different scenarios for how the company data can get exposed:

xoxb – Custom Bot Token

A xoxb-token (which has the prefix xoxb) is created by the "Custom Bot"-page for each team. This token is detached from the creator, it is marked as a bot and behaves like a separate user. You cannot restrict this token to disable a certain API endpoint. It does have one significant restriction though, the search.all API will respond with: `Error: user_is_bot`. However, since it still has access to all channel's messages, all data can be retrieved using this token.

xoxp – Private Token

The xoxp-token (prefix xoxp) can be generated from the OAuth Test Token-page. This token is exactly like having the complete username and password for the user. Even for a user with two factor authentication enabled, you can still access Slack with nothing else but this token.

This token can use the search.all-API, being able to search not only for channel messages and files, but also in the current user's internal messages and groups.

There are additional types of tokens, but these two types are the ones being leaked the most.

We found 626 private tokens and 879 bot tokens. On the page where you generate the Private Tokens, this message is shown:

> ⚠ **Test tokens are just for you.** Treat them as you would a password. Never share test tokens with other users or applications.

On the Custom Bot page on Slack, there's no indication at all that the token should be kept secret, but looking at the data, that message on the Private Token page doesn't really make any difference.

When we identified companies running responsible disclosure, with security contact or an email for the owner of the repository, we got mixed results as you see here:

1:29 AM	so, there are publicly available slack tokens to .slack.com
1:30 AM	oh
1:30 AM	what
1:30 AM	where?
1:30 AM	https://github.com/ ?tab=repositories
1:30 AM	for fucks sake
1:30 AM	been there for 6 days
1:30 AM	and I can see everything
1:30 AM	oh fuck
1:36 AM	I'm getting them revoked now
1:36 AM	but i won't be able to ack the report until tomorrow morning gmt
1:36 AM	I just meant
1:36 AM	no worry
1:36 AM	send me a thing so I can give you cash
1:42 AM	sent

Apr 18 (3 days ago)

to me

Hey Frans,

I wasn't aware that this kind of information was exposed even with private repos. I've taken steps to remediate…the exposure is gone. Thanks for the heads up!

Frans Rosén <frans@detectify.com> Apr 18 (3 days ago)

to

Great !

I actually think the repo was accidentally put as public as it showed up using GitHub's search API, but great that you removed it! Please remember to revoke the token at api.slack.com also and generate a new one from there! :)

Cheers,
Frans

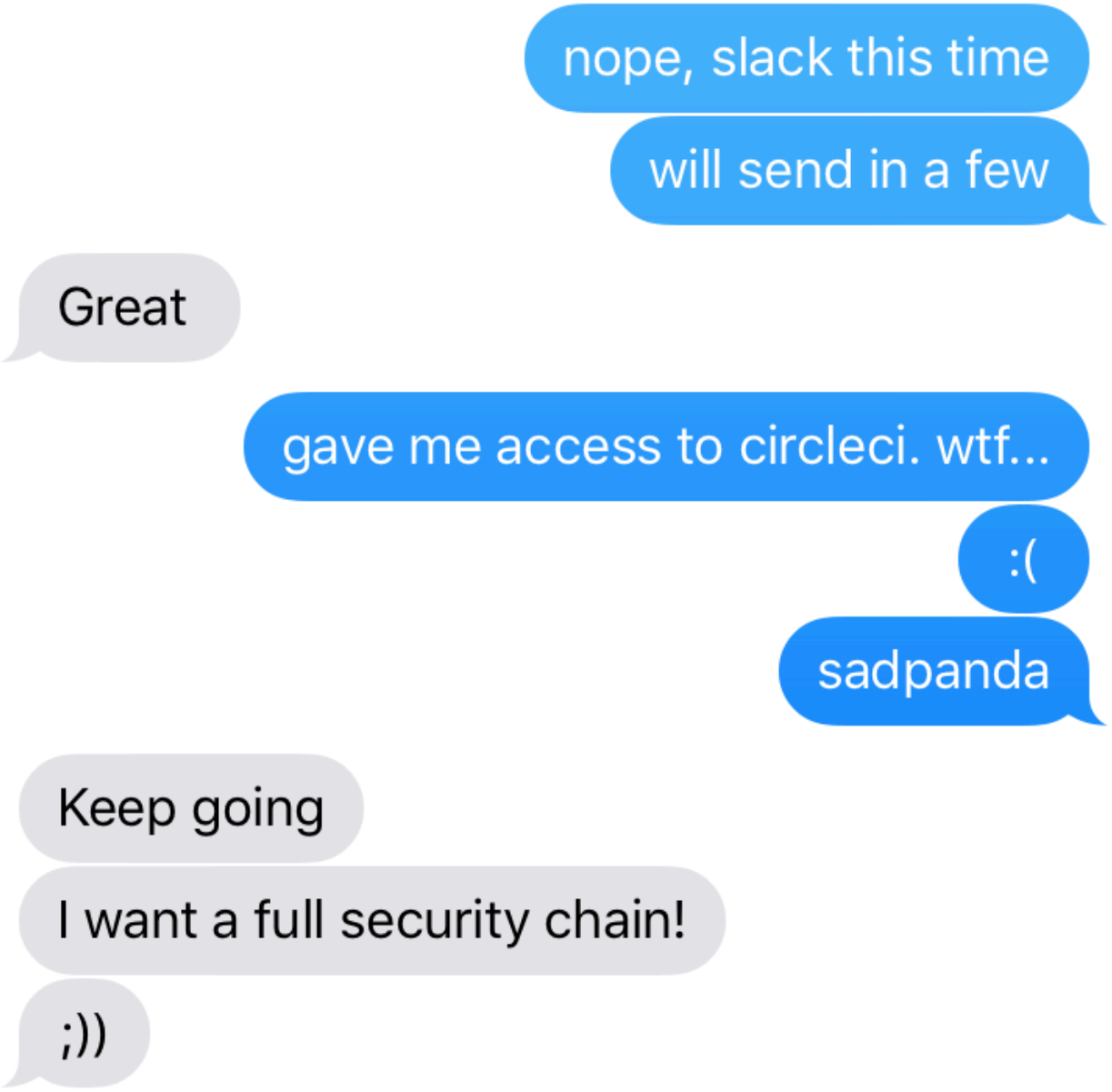
Apr 18 (3 days ago)

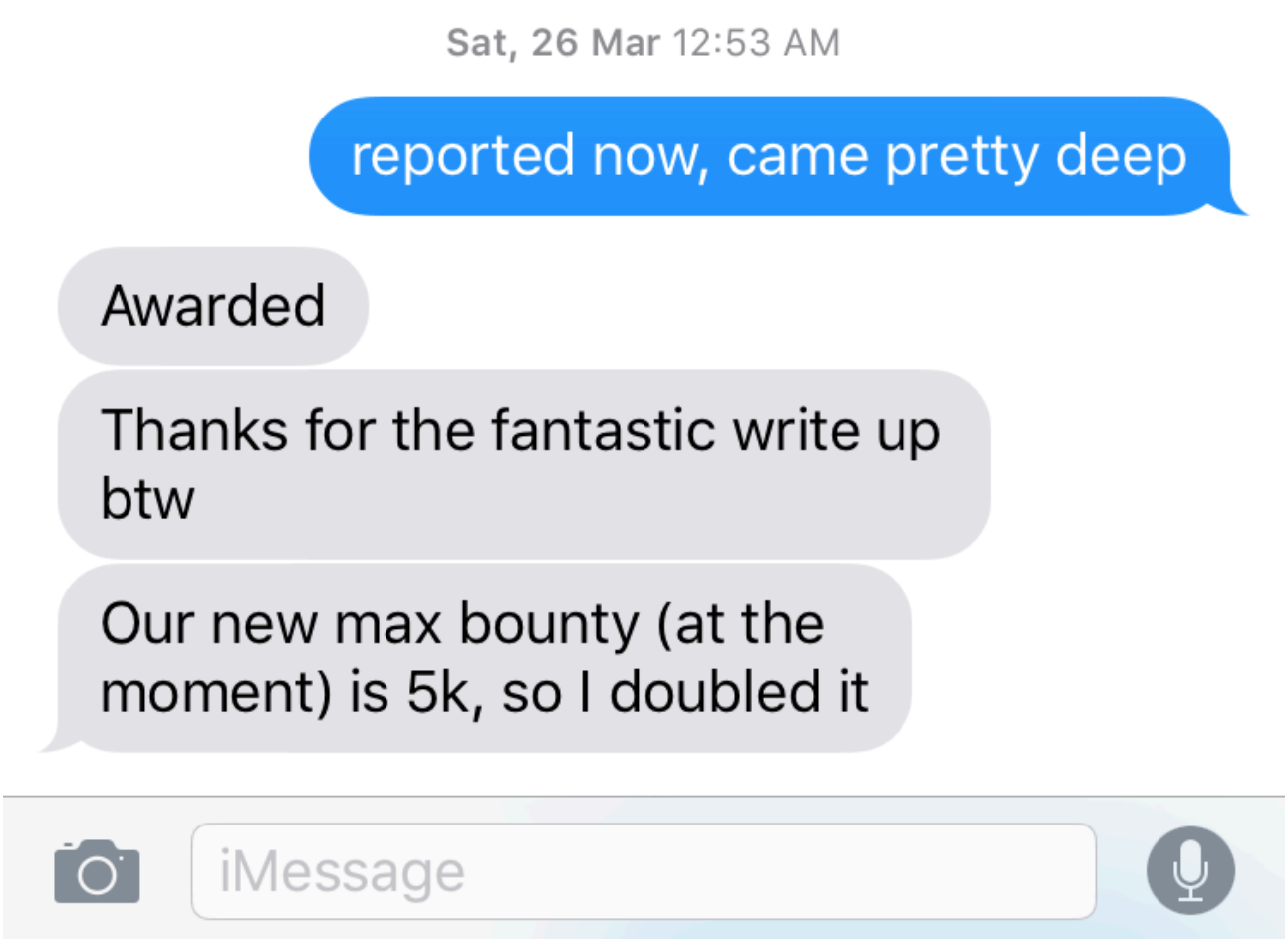
to me

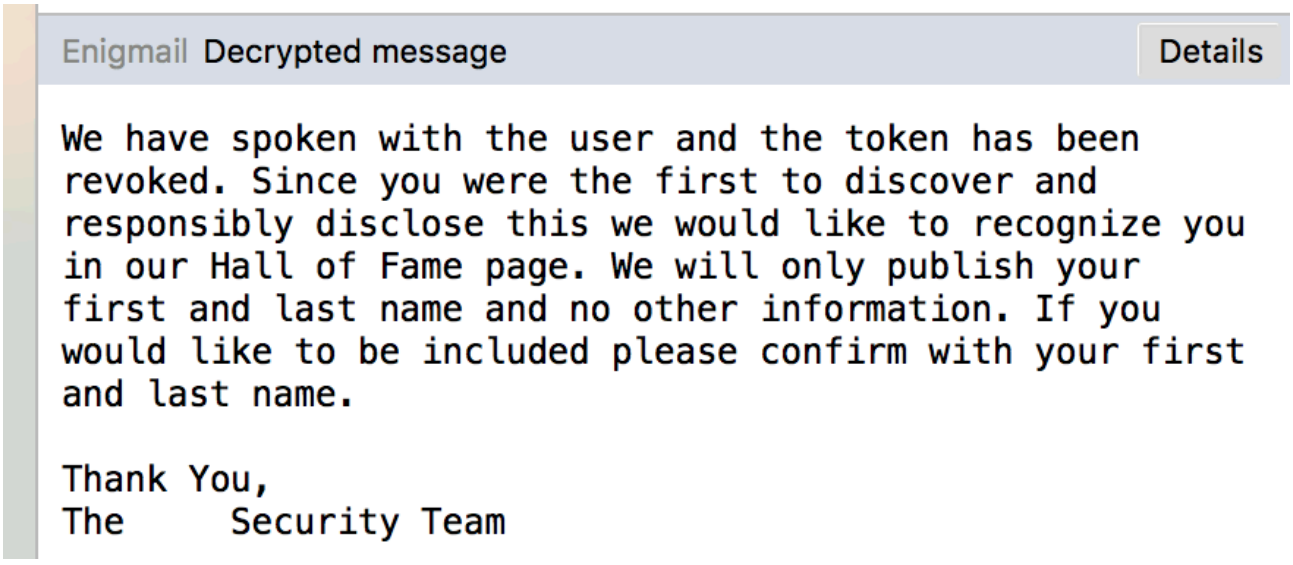
Frans,

Yeah, I just realized that I accidentally set it up as public.

Thanks!


nope, slack this time
will send in a few
Great
gave me access to circleci. wtf...
:(
sadpanda
Keep going
I want a full security chain!
;)))





posted a comment.

Hi, once again thank you for the report. I've attached a copy of our NDA for your to look at. Our legal folks would like you to sign it before we issue your bounty. The bounty will be $5,000/USD. Please review the NDA and let me know if you have any questions or concerns.

What we found:

Database credentials, login to continuous integration platforms and internal services, private messages to the token owner and files with passwords. We also concluded from the internal communication inside Slack teams, that people tend to be really sloppy with passing credentials in general.

What happened next?

We reported this to Slack as “The sad state of Slack tokens on GitHub” on Mar 26th. The report was closed as informative by Slack on Apr 18th. During the discussions we mentioned different ways for Slack to actually be proactive around the cause of this. Some of the things was to be able to restrict access per API, being able to revoke the API-key using the API itself and to automatically revoke the tokens that ended up on GitHub.

Slack believe that it is foremost the responsibility of the users not to be sharing their tokens around – and are also clear with stating this on their website when a token is created.

The problem though, is what to do when people are obviously sharing this data publicly online daily.

In their last response they said;

“Thanks for the additional tokens. We’re proactively looking for tokens ourselves now, and reaching out to customers to let them know when we’ve disabled tokens and where we found them. We’ll deactivate these in the next batch.”

We haven’t been able to confirm this is actually the case yet but it sounds like a great plan.
Update: We received the message sent by Slack to teams with leaked tokens:

Hi there,

We recently became aware that one or more authentication tokens for your Slack team, xxx.slack.com, have been posted publicly on the Internet. This can happen when a developer or other member of your organization posts their private token on a public forum or website (usually a code-sharing website like GitHub).

To help protect your team's information, we're taking the precautionary step of permanently disabling the affected tokens on your behalf. This may cause some disruption for you or your team, such as a bot being disabled or an integration no longer working - and for that, we're sorry. However, maintaining the security of your team and privacy of your communications is important to us, even if it causes some inconvenience.

Here are the details of the tokens we've disabled:
xoxp-xxxxxxxx-xxxxxxx-xxxxxxxx-xxxx (Created by xxxxxx)

If your team was relying on these tokens for bots or integrations, please read https://get.slack.help/hc/en-us/articles/215770388 for information on how to issue new authentication tokens. You can control which team members have the ability to create tokens at https://xxx.slack.com/admin/settings#integrations.

Finally, if you have additional questions, you can reply directly to this notification - our support team is standing by and ready to help.

-The team at Slack

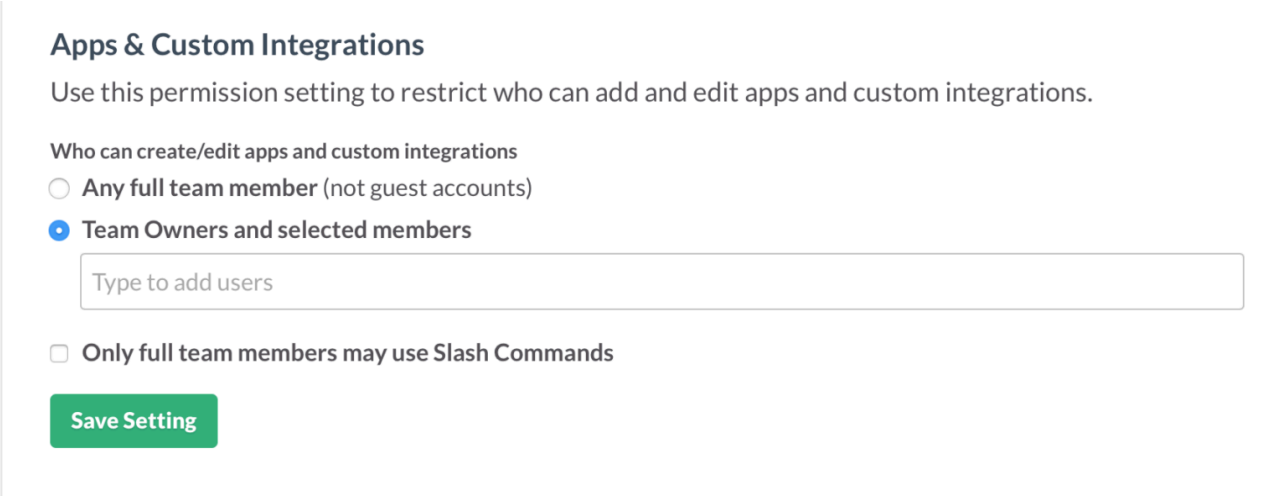
Best Practice

NEVER COMMIT CREDENTIALS INSIDE CODE. EVER.

The first thing you should do is to create environment-variables inside a file and ignore that file from the code repository from start. Also, public due diligence of your company: What type of public repositories do you have? How does the commit history of them look like? Do they have a bunch of legacy in them? Could there be any credentials forgotten inside the commit history? Are all developers in our company able to create private repositories?

When passing credentials internally, use services like onetimesecret.com to place passwords in a only-visible-once out-of-context. Sending the URL to the secret together with the username in the message will be an improvement compared to today, since the viewer of the secret can only watch it once, and if onetimesecret.com is hacked, no one will know what the password belongs to.

Also, Slack has some settings that could be great to enable. The important one is disabling the ability for all users to create integrations and tokens:



You will find this setting over at Admin Settings in Slack.

Last words

GitHub is full of sensitive data. Slack just made it really simple to search for their tokens due to how they are formed. We hope that this advisory might help people realize how big impact getting these tokens exposed really is.

Also, if we can help limit the amount of robots in the world somehow, the Skynet hopefully won't happen that soon.

(We love robots, but not exploited ones)
