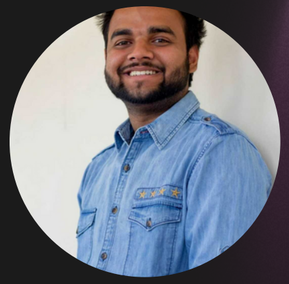
How We Got Into A Unicorn's Private Codebase

The Team



Arshit Jain

FULL STACK
ENGINEER

Ashikka Gupta

SECURITY RESEARCH +
TECHNICAL WRITING
INTERN

Mannan Goyal

CYBER SECURITY
ANALYST INTERN

Agenda

Part 1

WHAT EXACTLY HAPPENED WITH THE UNICORN?

Part 2

LARGE-SCALE STUDY OF OVER 1M+ APPS

Part 3

HOW DO WE AVOID THESE MISTAKES AND THEIR IMPACT?

Part 1

The Twitch Leak 2021

6,000

Internal Git repositories

200 GB

worth of data

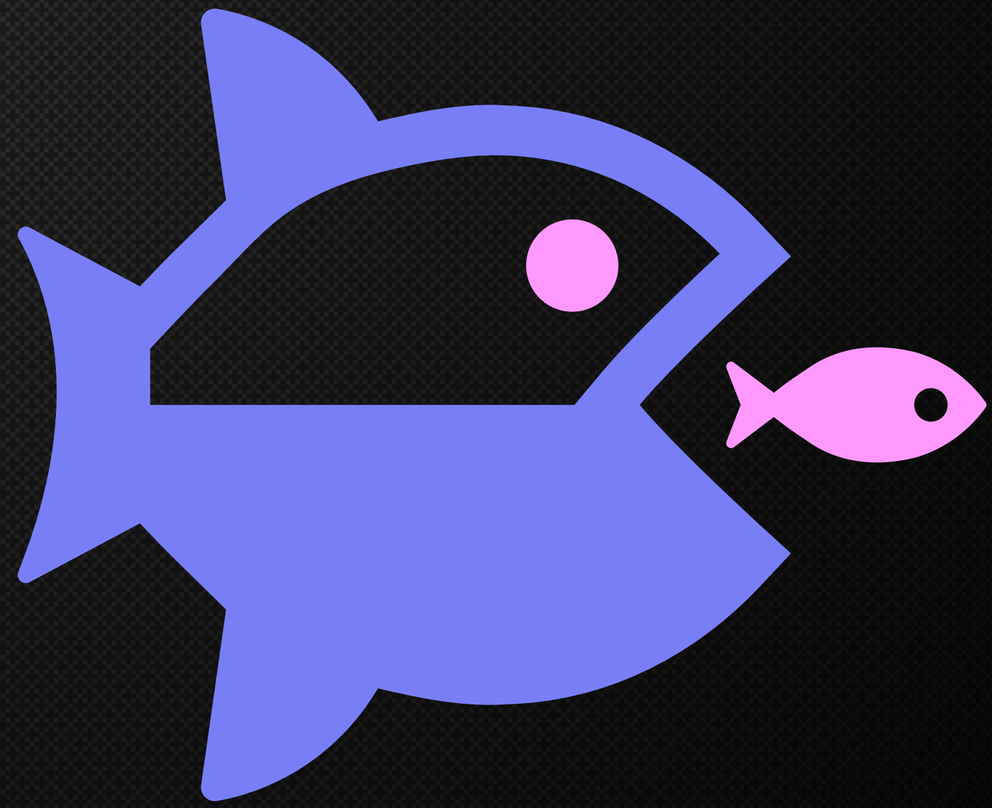
3,000,000

Documents

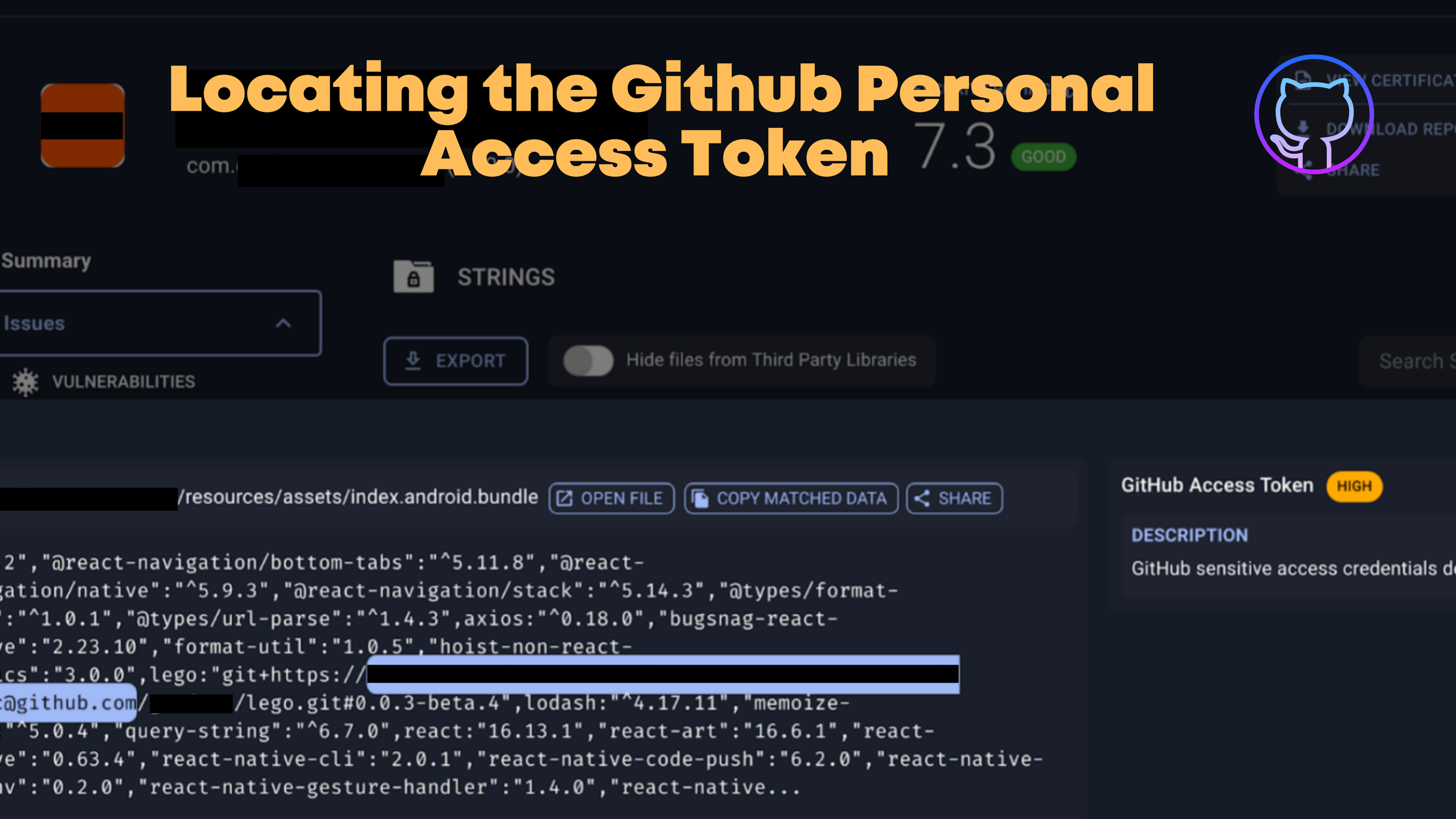
The Big Fish

We present to you the biggest of our findings.

- **$120M** worthy unicorn
- **10M+** downloads on Play Store



Locating the Github Personal Access Token



What's a PAT Token and what can it do?

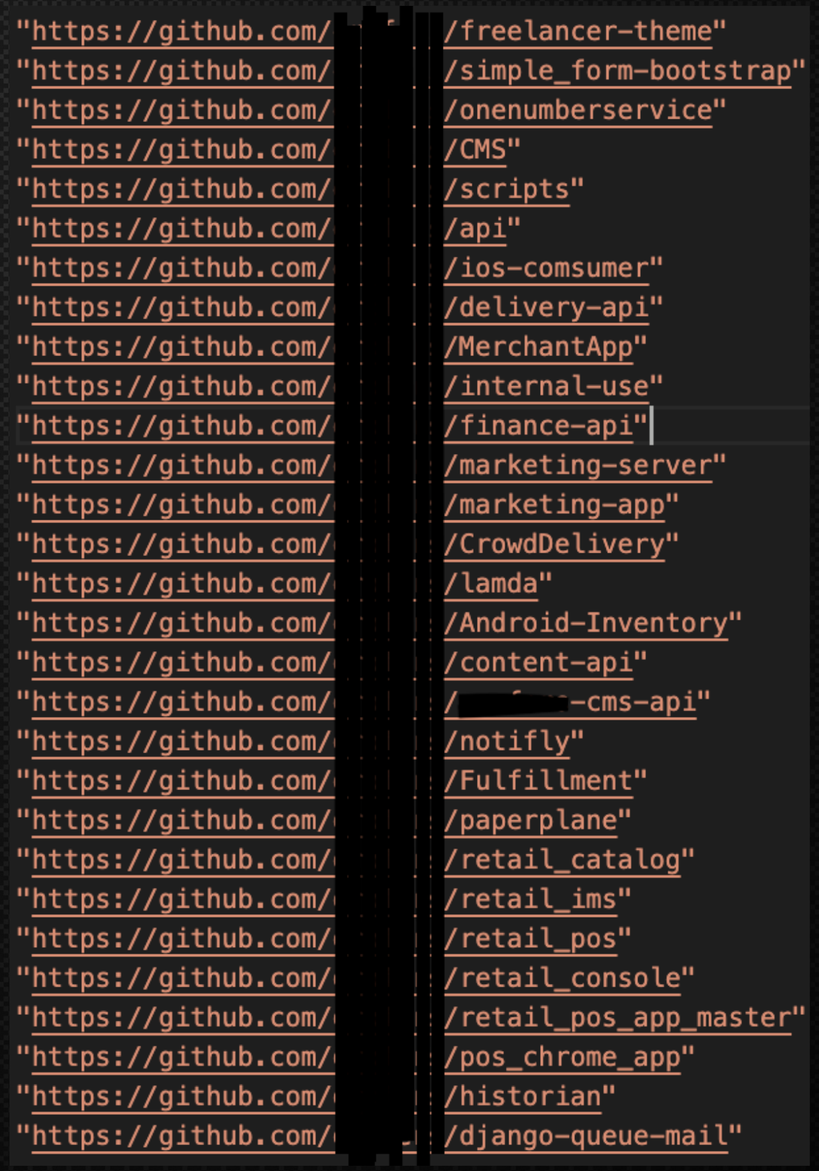
```
arshitjain@Arshits-MacBook-Pro ~ % curl --head -H "Authorization:                                        "https://api.github.com/user" | grep -Fi x-oauth-
scopes
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0  1658    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
x-oauth-scopes: admin:enterprise, admin:gpg_key, admin:org, admin:org_hook, admin:public_key, admin:repo_hook, delete:packages, delete_repo, gist, notifications, r
epo, user, workflow, write:discussion, write:packages
access-control-expose-headers: ETag, Link, Location, Retry-After, X-GitHub-OTP, X-RateLimit-Limit, X-RateLimit-Remaining, X-RateLimit-Used, X-RateLimit-Resource, X
-RateLimit-Reset, X-OAuth-Scopes, X-Accepted-OAuth-Scopes, X-Poll-Interval, X-GitHub-Media-Type, X-GitHub-SSO, X-GitHub-Request-Id, Deprecation, Sunset
arshitjain@Arshits-MacBook-Pro ~ %
```

Scope repo	**Gives full access to private repositories**

Verifying The Loot

```
blackjack@DESKTOP-EA82A1K:~$ curl -H "Authorization: token [redacted]" "https://api.github.com/user/repos"
[
  {
    "id": 427507358,
    "node_id": "R_kgDOGXs-ng",
    "name": "[redacted]",
    "full_name": "[redacted]",
    "private": true,
    "owner": {
      "login": "[redacted]",
      "id": 92807706,
      "node_id": "O_kgDOBYgiGg",
      "avatar_url": "https://avatars.githubusercontent.com/u/92807706?v=4",
      "gravatar_id": "",
      "url": "https://api.github.com/users/[redacted]",
      "html_url": "https://github.com/[redacted]",
      "followers_url": "https://api.github.com/users/[redacted]/followers",
      "following_url": "https://api.github.com/users/[redacted]/following{/other_user}",
      "gists_url": "https://api.github.com/users/[redacted]/gists{/gist_id}",
      "starred_url": "https://api.github.com/users/[redacted]/starred{/owner}{/repo}",
      "subscriptions_url": "https://api.github.com/users/[redacted]/subscriptions",
      "organizations_url": "https://api.github.com/users/[redacted]/orgs",
      "repos_url": "https://api.github.com/users/[redacted]/repos",
      "events_url": "https://api.github.com/users/[redacted]/events{/privacy}",
      "received_events_url": "https://api.github.com/users/[redacted]/received_events",
      "type": "Organization",
      "site_admin": false
    },
    "html_url": "https://github.com/[redacted]/accessibility-audit",
    "description": null,
```

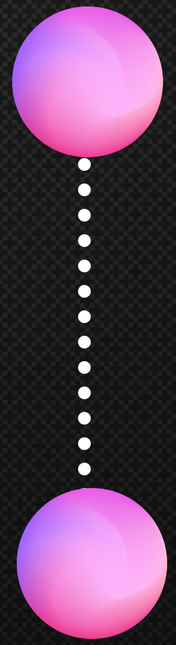
And this how we got access to the Unicorn's codebase...



Count the mistakes

How were we able to do this?

There were 2 major mistakes on the part of the developers:



MISTAKE 1

Hardcoding the GitHub PAT token in the source code.

MISTAKE 2

Giving excessive scope to the token which can be used by anyone for exploitation

Part 2

We built our own security search engine

Step 1

COLLECTION OF MOBILE APPS

Step 2

DECOMPILING APPS

Step 3

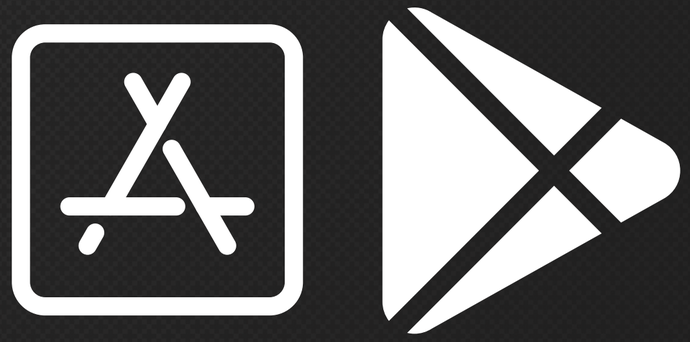
BUILDING REGEXES

Step 4

FIND REGEX MATCHES ON LARGE SCALE

Collection of Mobile Apps

User submissions that included apps user uploaded apps.



All android app stores over internet

Decompiling Apps

Open Source Android Decompilers, like JadX

```
package uk.co.ribot.androidboilerplate;

import android.app.Application;
import android.content.Context;

public class AndroidApplication extends Application  {

    ApplicationComponent mApplicationComponent;

    @Override
    public void onCreate() {
        super.onCreate();

        if (BuildConfig.DEBUG) {
            Timber.plant(new Timber.DebugTree());
            Fabric.with(this, new Crashlytics());
        }
    }

    public static AndroidApplication get(Context context) {
        return (AndroidApplication) context.getApplicationContext();
    }

    public ApplicationComponent getComponent() {
        if (mApplicationComponent == null) {
            mApplicationComponent = DaggerApplicationComponent.builder()
                    .applicationModule(new ApplicationModule(this))
                    .build();
        }
        return mApplicationComponent;
    }
```

THE TOUGHEST OF IT ALL

Github

```
(?i)github(.{0,20})?(?-i)['\"][0-9a-zA-Z]{35,40}
```

Google API Key

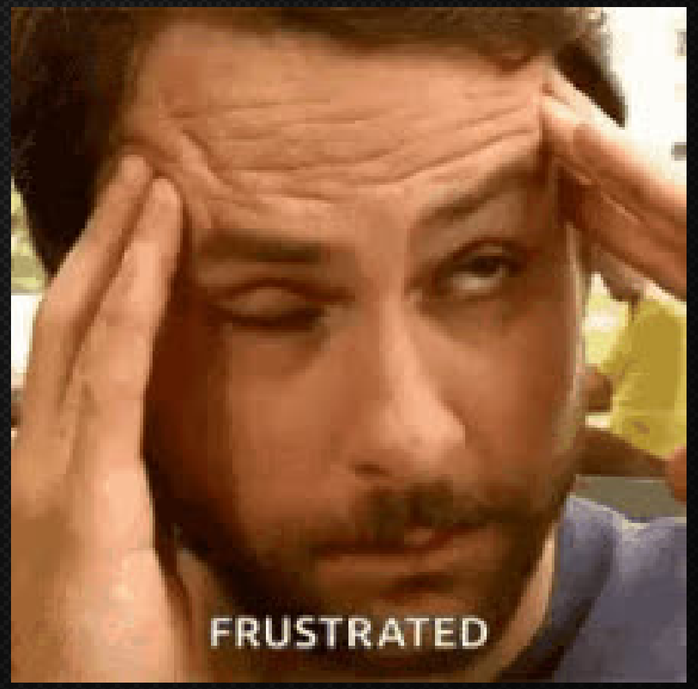
```
AIza[0-9A-Za-z\\-_]{35}
```

Google Cloud Platform API Key

```
(?i)(google|gcp|youtube|drive|yt)(.{0,20})?['\"][AIza[0-9a-z\\-_]{35}]['\"]
```

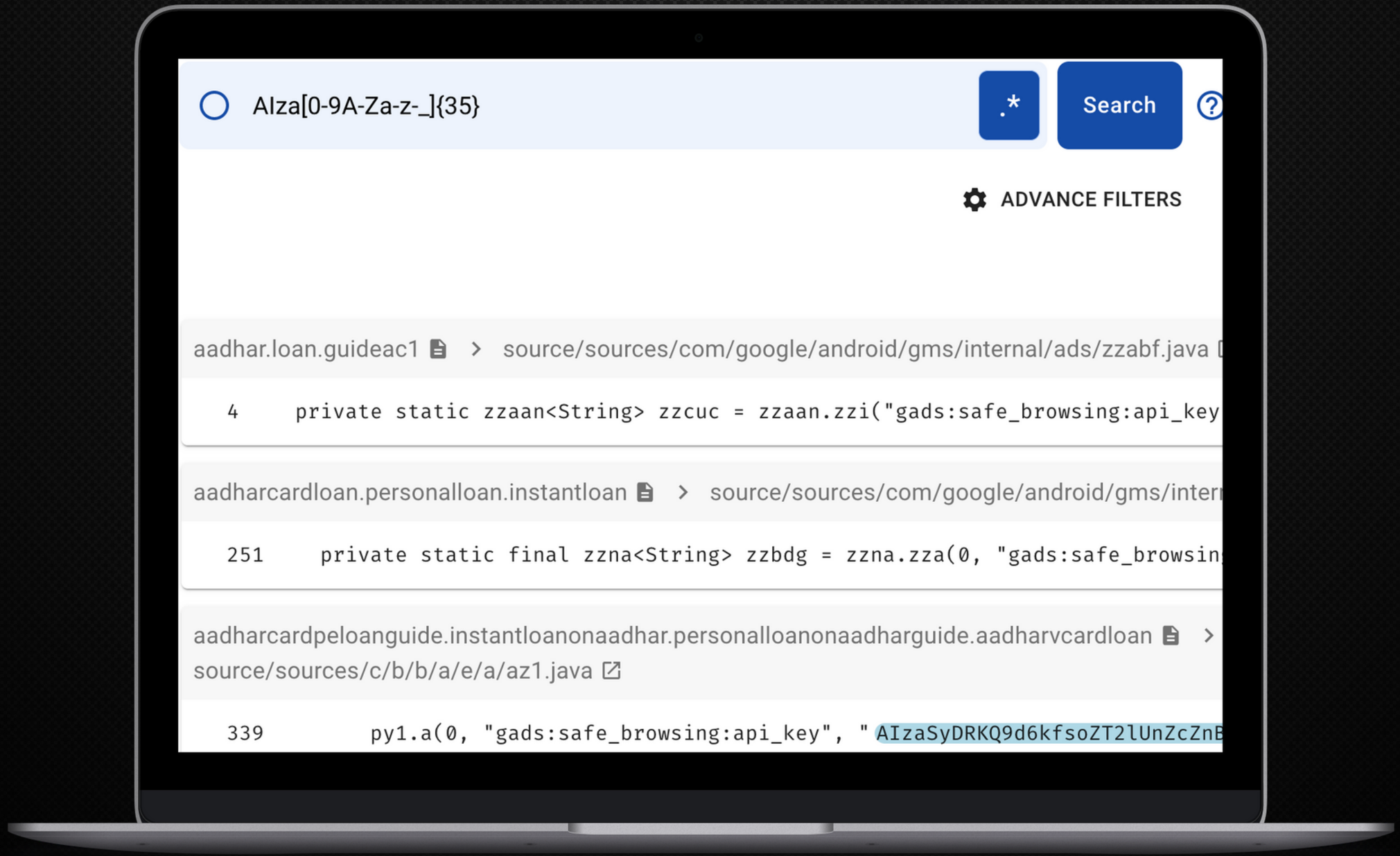
Google Drive API Key

```
AIza[0-9A-Za-z\\-_]{35}
```



Another resource that was helpful in particular was the RegHex repository by l4yton,trufflehog. We did proper regex testing by adding all the keys in an android application and testing that android application for checking for keys and false positives.

RegEx Matches on an extensive scale



Part 3

And the story doesn't here...

More GitHub PATs found

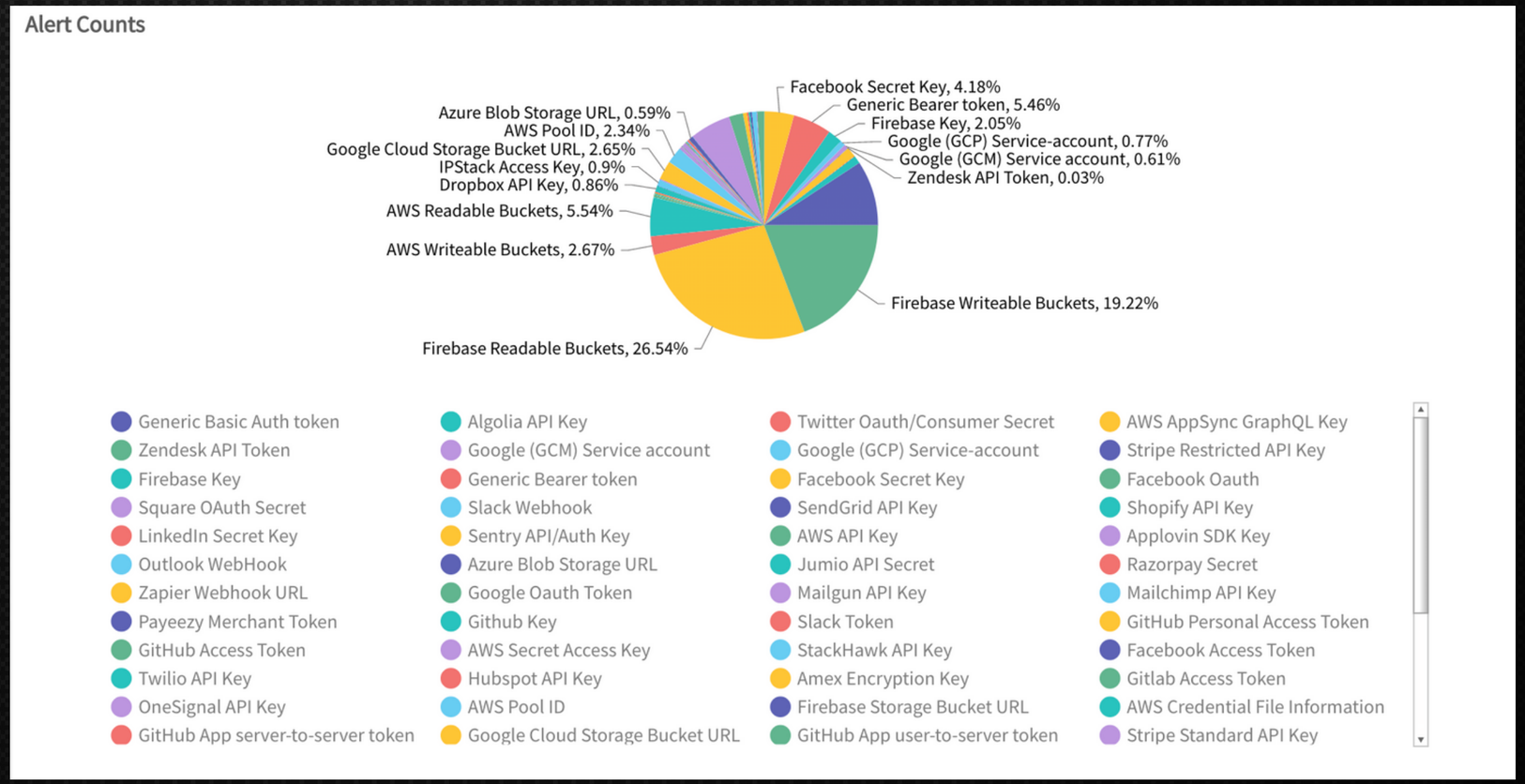
We found 159 private repositories from 151 Github token in different mobile apps. These apps had downloads ranging from 100 to over 10M on playstore.

The impact is as follows:

- Source code might leak secrets like database config details, static keys and cloud environment variable.
- Financial losses to companies
- Brand Confidence might also go down

1.6M+ HARDCODED SENSITIVE TOKENS

INT, FLOAT, DOUBLE.. I MEAN LET'S TALK NUMBERS



Impacts of Secret Key Leaks

To make you guys feel the impact of the leaks here we have two categories of of leaks that we have found in the past as examples-

1. **Email Automation**

2. **Payment Processing**

Email Automation Tools

Following are some of the impacts of credentails leaked in email automation tools like Sendgrid, Mailchimp,Mailgun etc.

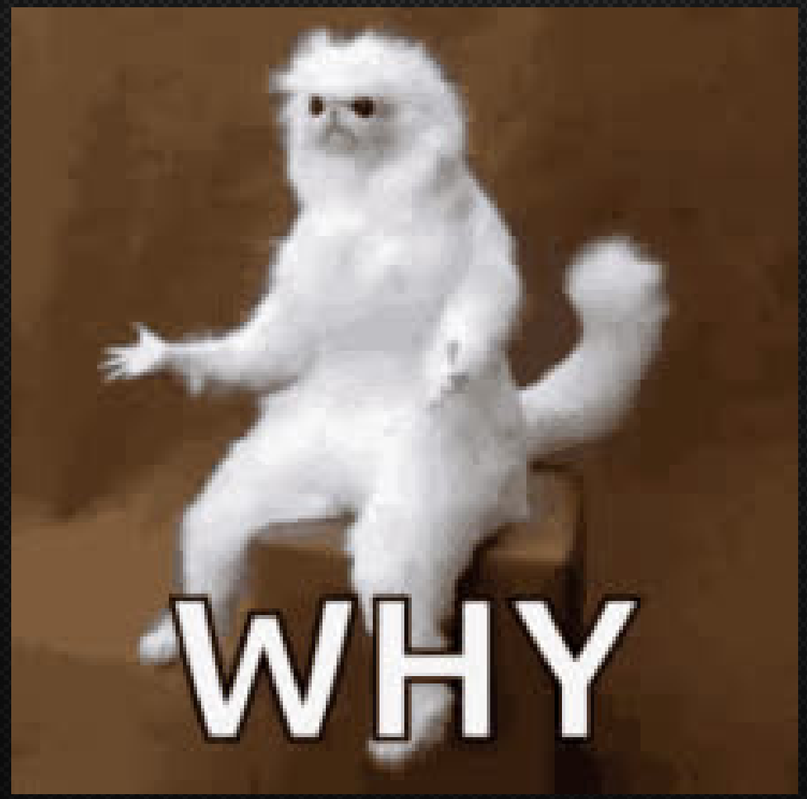
- Send phishing emails.
- Access to users personal information including name,email,phone number.

Payment Processing Tools

Following are some of the impacts of credentails leaked in payment processing tools like Razorpay, Stripe etc.

- Initiate Refund to bank account
- Access to all companies payment transaction details .

Why do developers do this?



Problems Faced by Android Developers

1

Security Pipeline

Pain of setting up a proper mobile app security testing pipeline while development.

2

Awareness

Lack of awareness on the scope/impact of the Hardcoded secret.

3

Budgeting

Companies not spending much on doing proper security testing on mobile apps - compared to web apps.

Solutions

1

Scoping

Most services allow developers to allowing only certain activities on an API key, so that even if an attacker gets their hands on it they will still have limited access.

2

ENV

Make use of environment variables to store API Keys instead of hardcoding them.

Solutions

3

Git Hooks

Make use of Git Hooks like Husky in projects to prevent people form commiting sensitive information onto platforms like GitHub, GitLab etc.

4

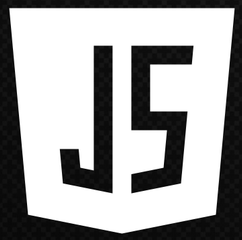
Testing Pipeline

End to end mobile app security testing pipeline while dev

Future Roadmap

Till now we only have conducted our research on Android mobile application, but we plan to scale up our research to a larger horizon.

Client-side Javascript



iOS mobile applications

iOS

Thank You