How to Hack OAuth

AARON PARECKI

@aaronpk

aaronpk.com

Senior Security Architect
at Okta

@oktadev

Micropub

W3C Recommendation 23 May 2017

Latest published version:
https://www.w3.org/TR/micropub/

Latest editor's draft:
https://micropub.net/draft/

Previous version:
https://www.w3.org/TR/2017/PR-micropub-20170413/

Editor:
Aaron Parecki

Repository:
Github
Issues
Commits

[Docs] [txt|pdf|xml|html] [Tracker] [WG] [Email] [Diff1] [Diff2] [Nits]

Versions: (draft-parecki-oauth-browser-based-apps)
00

Open Authentication Protocol — A. Parecki
Internet-Draft — Okta
Intended status: Best Current Practice — D. Waite
Expires: August 2, 2019 — Ping Identity
January 29, 2019

OAuth 2.0 for Browser-Based Apps
draft-ietf-oauth-browser-based-apps-00

Abstract

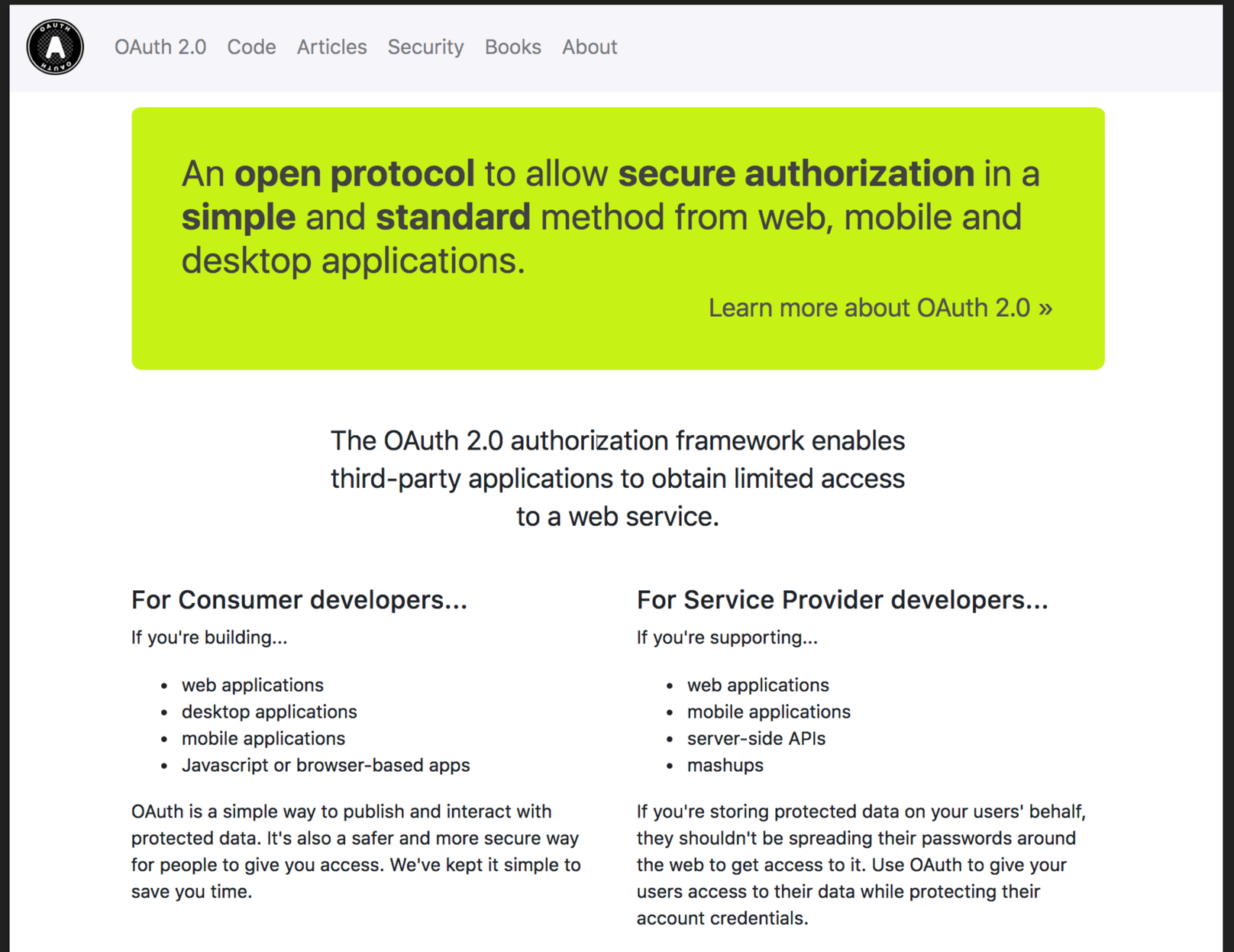
OAuth 2.0 authorization requests from apps running entirely in a browser are unable to use a Client Secret during the process, since they have no way to keep a secret confidential. This specification details the security considerations that must be taken into account when developing browser-based applications, as well as best practices for how they can securely implement OAuth 2.0.

Status of This Memo

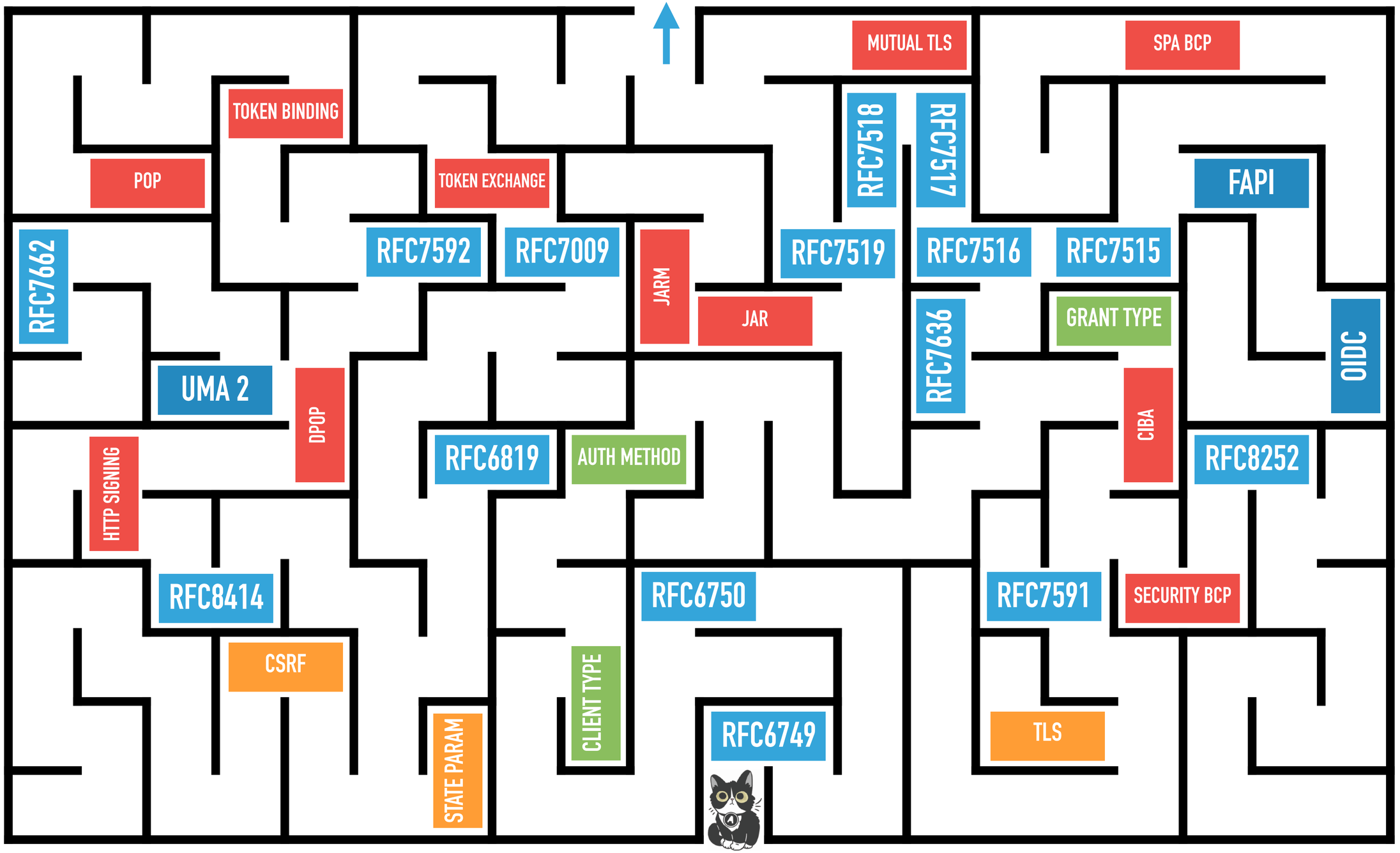
This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

@aaronpk

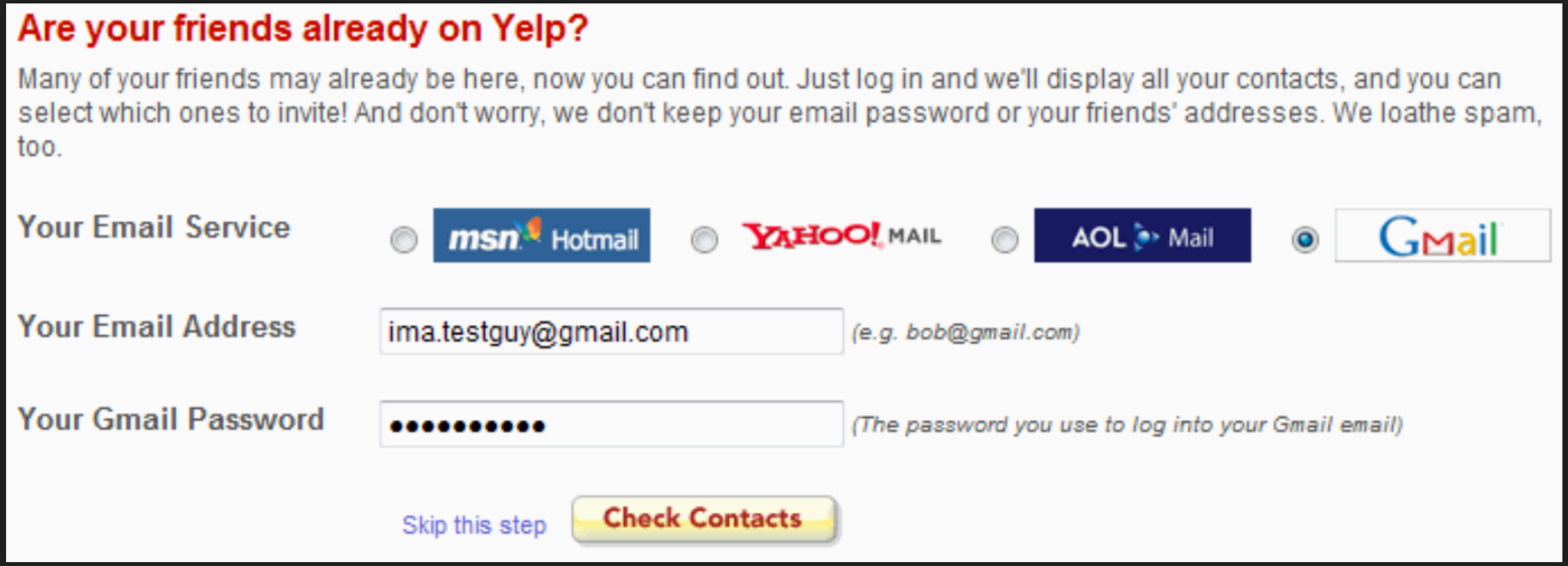
oauth.net


OAuth 2.0
Code
Articles
Security
Books
About
An **open protocol** to allow **secure authorization** in a **simple** and **standard** method from web, mobile and desktop applications.
Learn more about OAuth 2.0 »
The OAuth 2.0 authorization framework enables third-party applications to obtain limited access to a web service.
For Consumer developers...
If you're building...
web applications
desktop applications
mobile applications
Javascript or browser-based apps
OAuth is a simple way to publish and interact with protected data. It's also a safer and more secure way for people to give you access. We've kept it simple to save you time.
For Service Provider developers...
If you're supporting...
web applications
mobile applications
server-side APIs
mashups
If you're storing protected data on your users' behalf, they shouldn't be spreading their passwords around the web to get access to it. Use OAuth to give your users access to their data while protecting their account credentials.

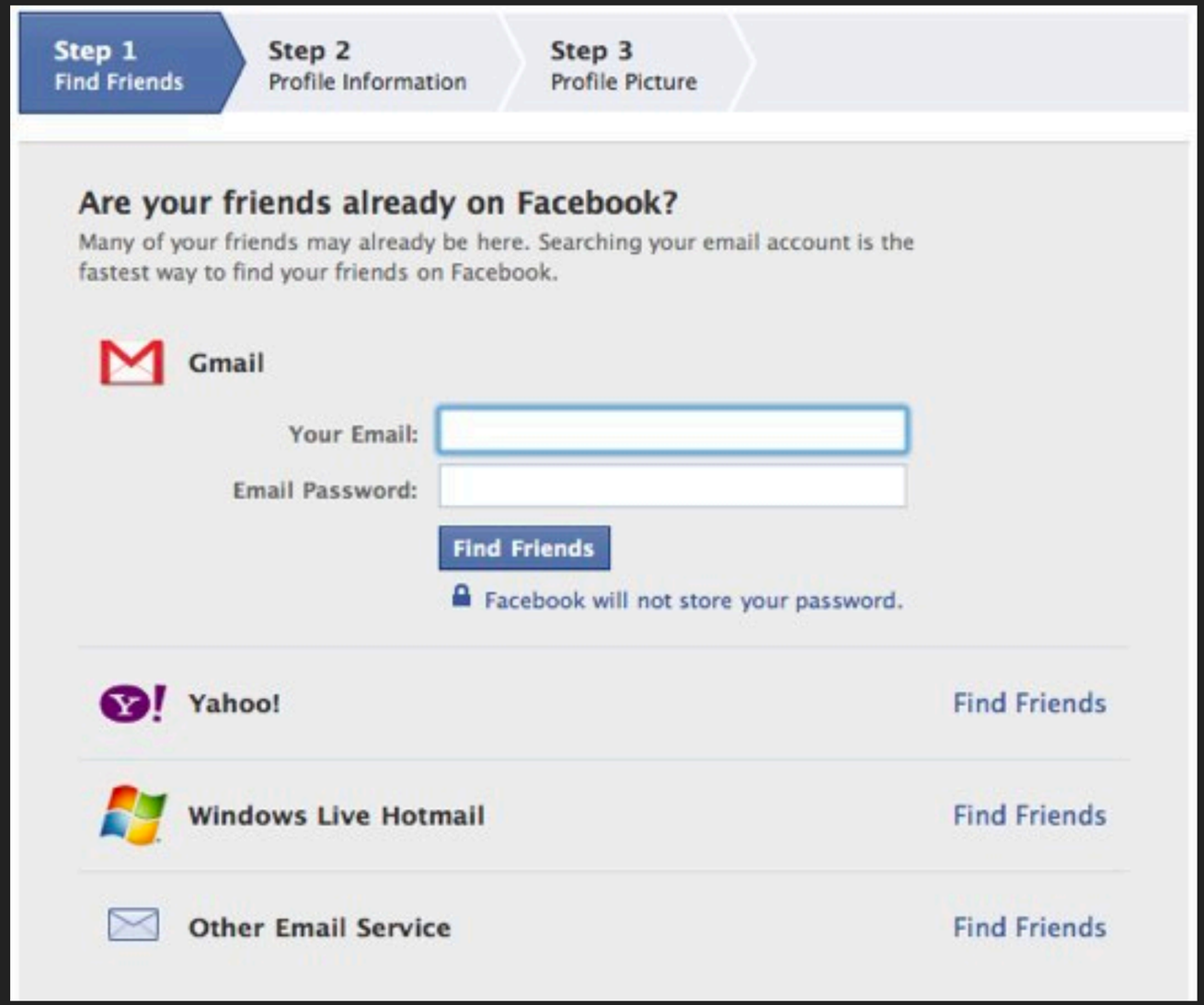
BUILDING YOUR APPLICATION



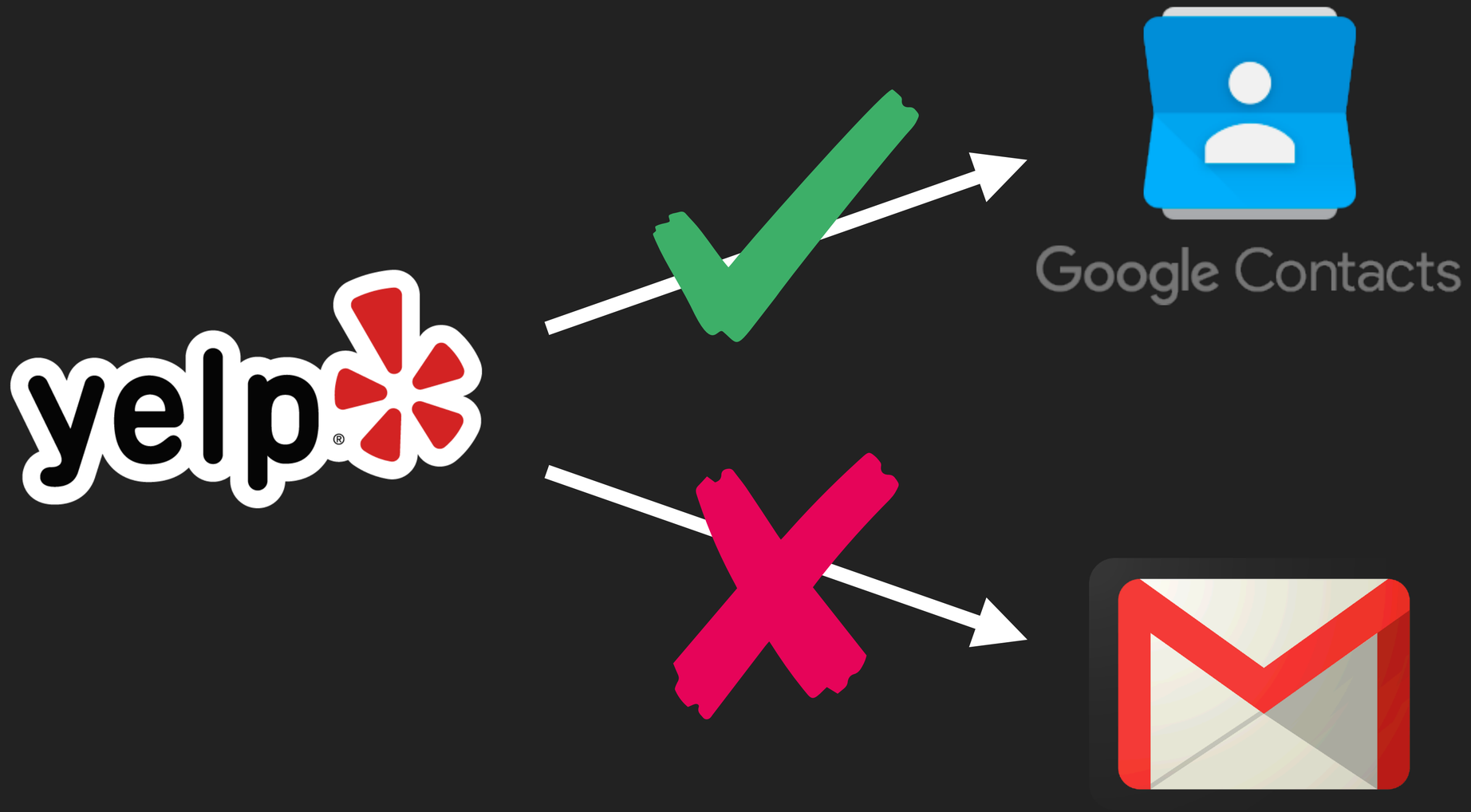
THE PASSWORD ANTI-PATTERN



THE PASSWORD ANTI-PATTERN



facebook.com ~2010

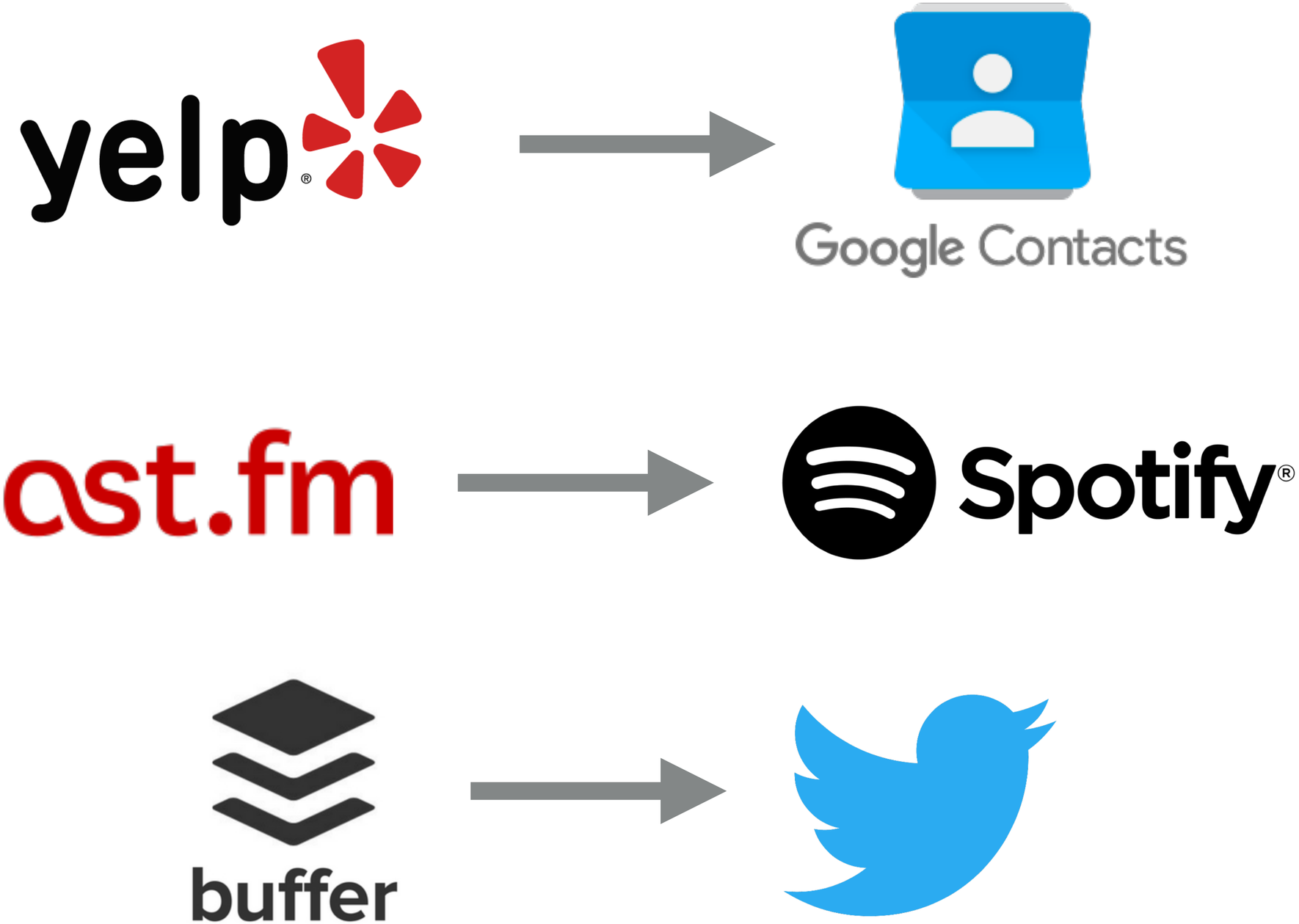

yelp
Google Contacts

so...

how can I let an app

access my data

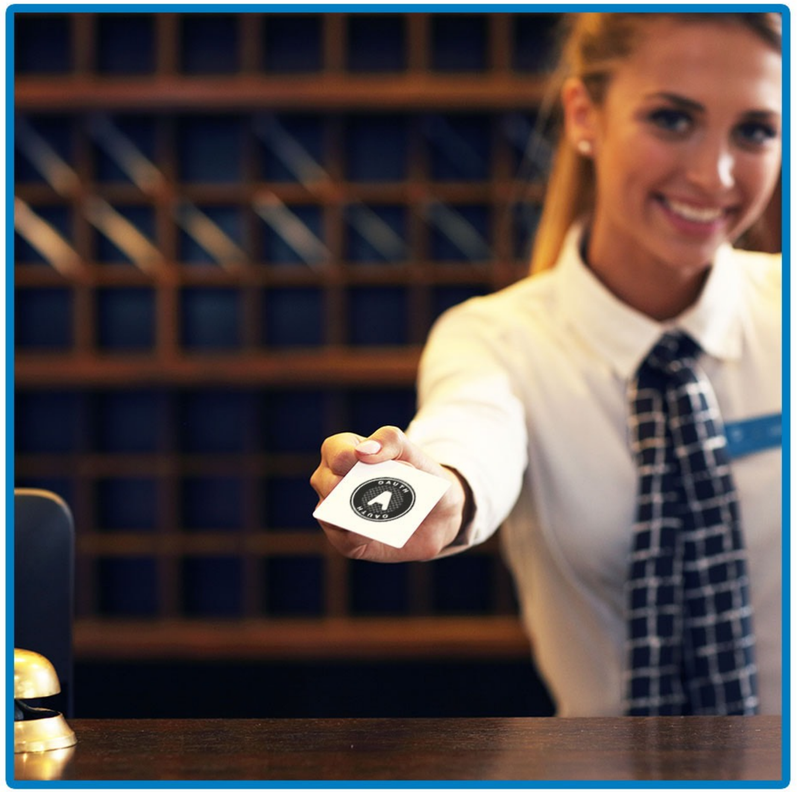
without giving it my password?


yelp
Google Contacts
last.fm
Spotify
buffer

```
POST /resource/1/update HTTP/1.1
Authorization: Bearer RsT5OjbzRn430zqMLgV3Ia
Host: api.authorization-server.com

description=Hello+World
```

A HOTEL KEY CARD, FOR APPS



Authorization Server

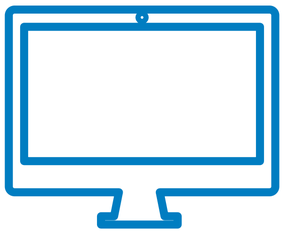
Access Token

Resource (API)

HOW OAUTH WORKS

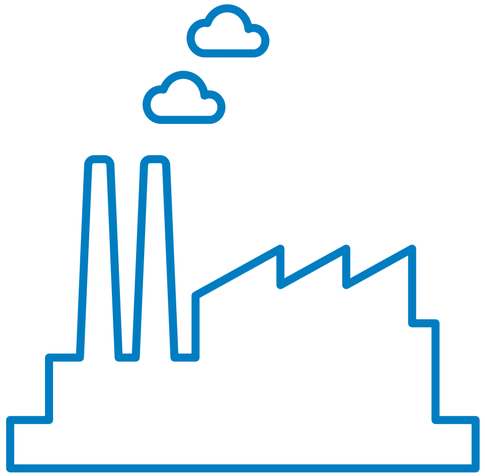
ROLES IN OAUTH

The User
(Resource Owner)



Device
(User Agent)

The Application
(Client)



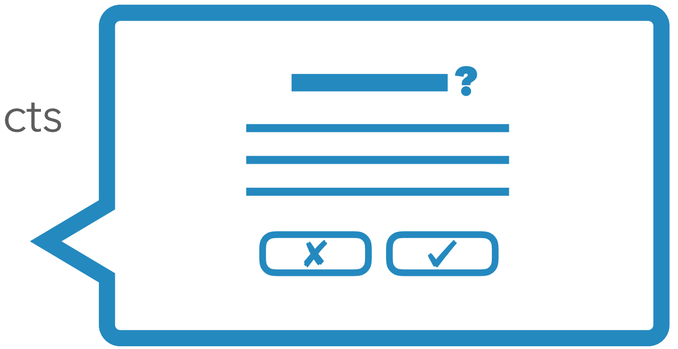
OAuth Server
(Authorization Server)
aka the token factory

API
(Resource Server)

User: I'd like to use this great app

App: Please go to the authorization server to grant me access

User: I'd like to log in to "Yelp", it wants to access my contacts



AS: Here is a **temporary code** the app can use

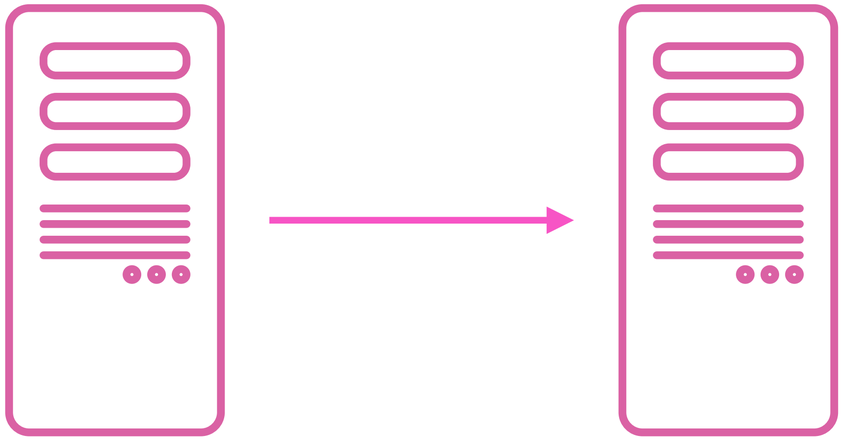
User: Here is the temporary code, please use this to get a token

App: Here is the temporary code, and my secret, please give me a token

AS: Here is an access token!

App: Please let me access this user's data with this access token!

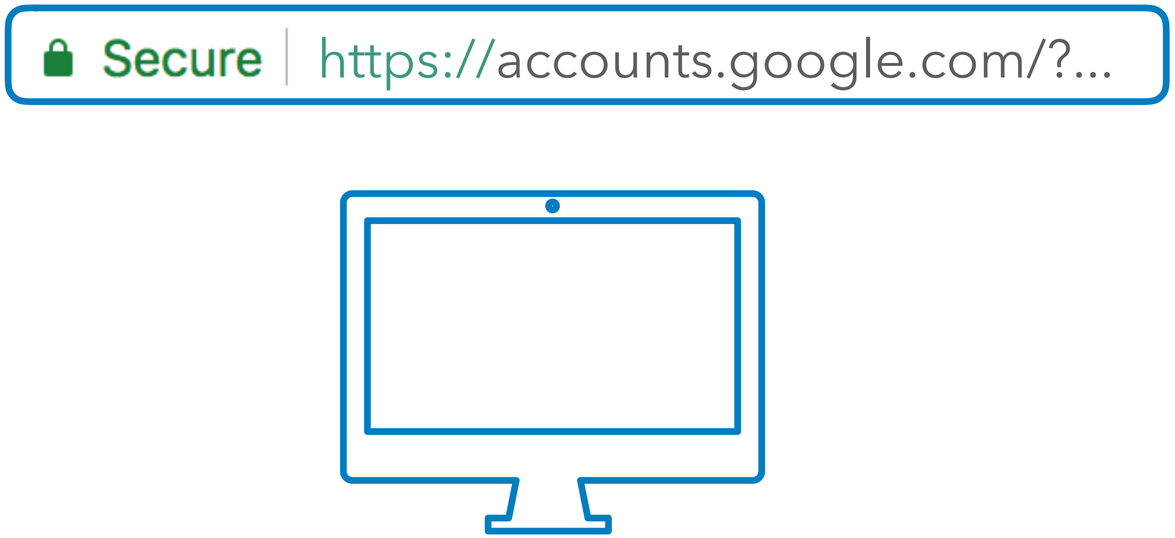
Back Channel



Sent from client to server

HTTPS request from client to server,
so requests cannot be tampered with

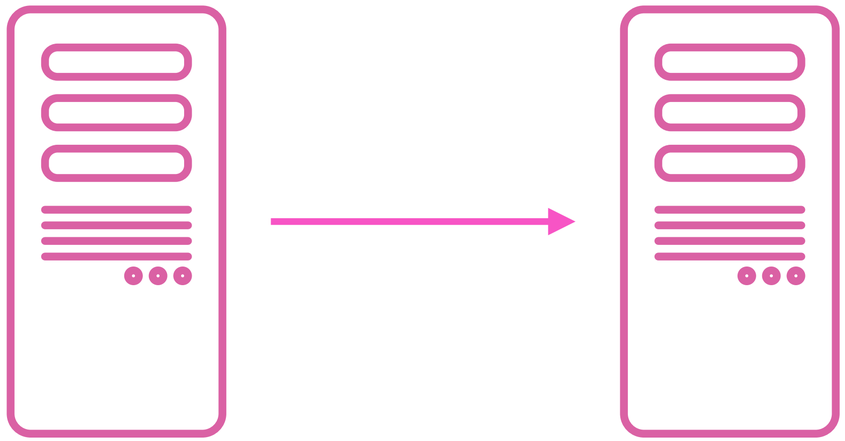
Front Channel



Passing data via the browser's address bar

The user, or malicious software,
can modify the requests and responses

Back Channel Benefits



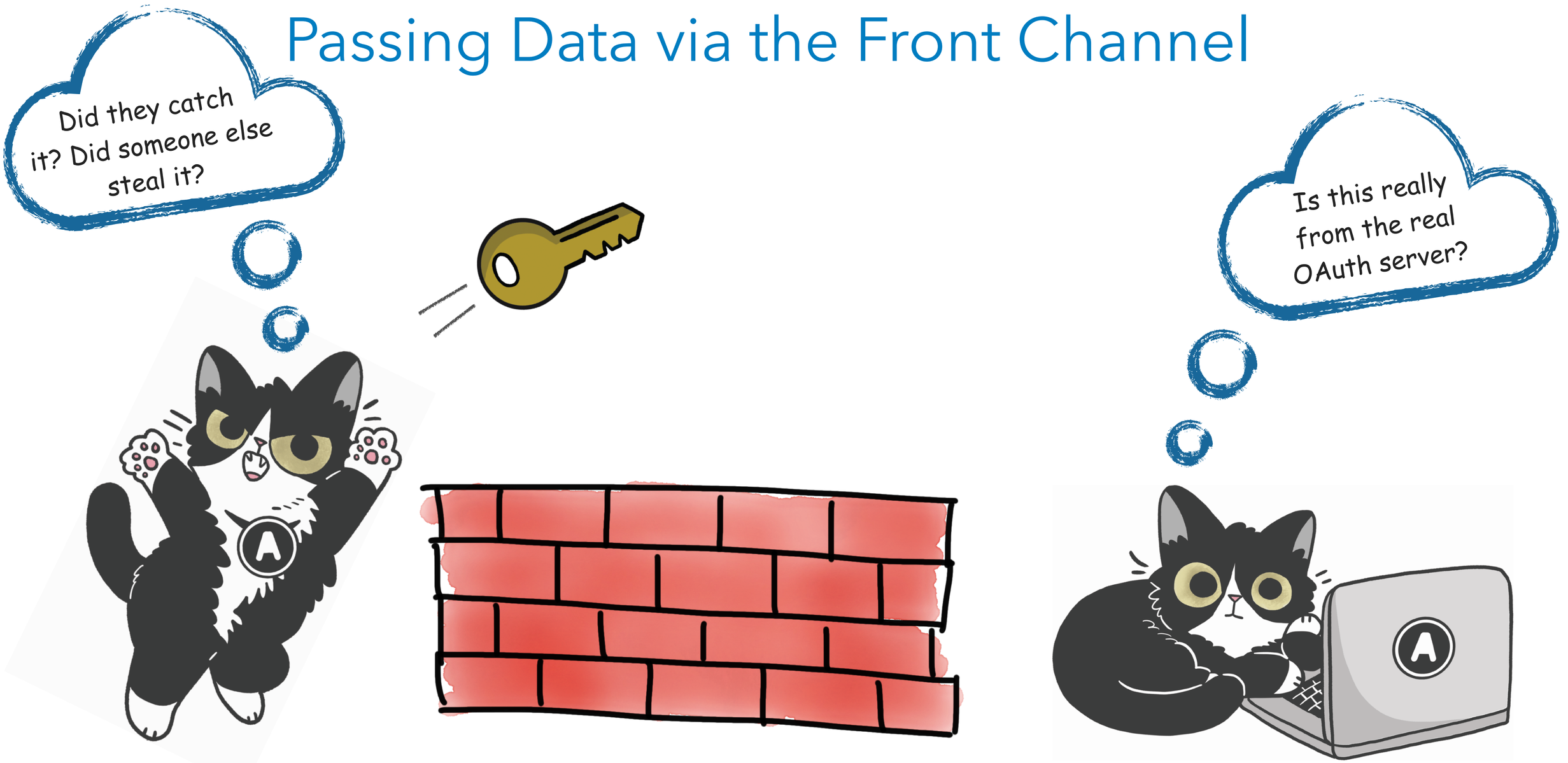
- The application knows it's talking to the right server
- Connection from app to server can't be tampered with
- Response from the server can be trusted because it came back in the same connection

Passing Data via the Back Channel

OAuth Server

OAuth Client

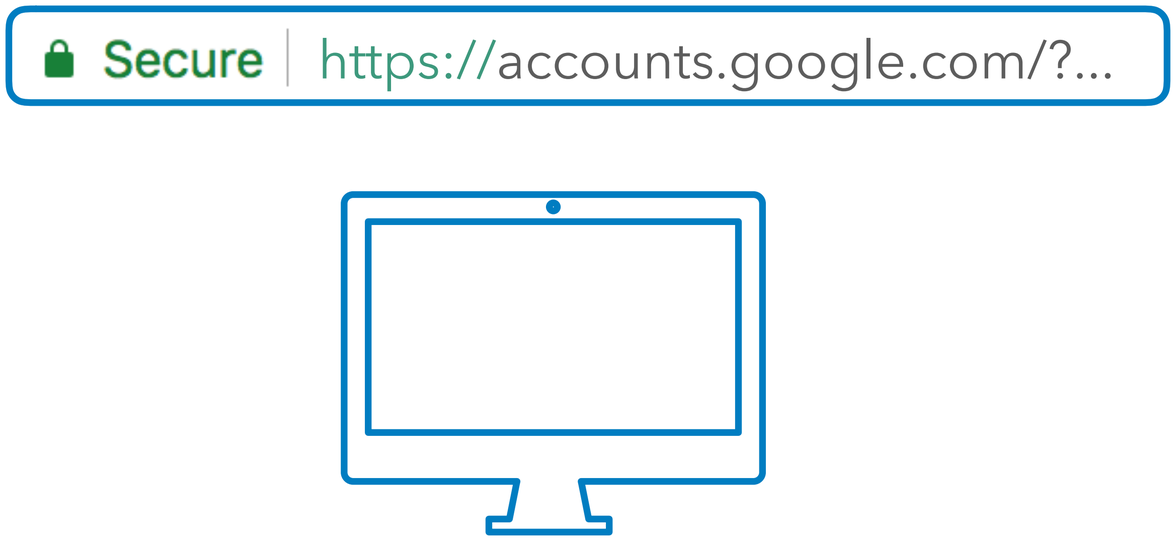
Passing Data via the Front Channel



OAuth Server

OAuth Client

Front Channel Benefits



- The user being involved enables them to give consent
- Enables easier two-factor authorization integration
- Doesn't require the receiver to have a publicly routable IP (e.g. can work on a phone)

THE HACKS

HOW TO HACK OAUTH

RFC 6749 Section 10

RFC 8252 Section 8

RFC 6819

draft-ietf-oauth-security-topics

[Docs] [txt|pdf] [draft-ietf-oaut...] [Tracker] [Diff1]

Updated by: 8252

Internet Engineering Task Force (IETF)
Request for Comments: 6749
Obsoletes: 5849
Category: Standards Track
ISSN: 2070-1721

The OAuth 2.0 Authorization Framework

[Docs] [txt|pdf] [draft-ietf-oaut...] [Tracker] [Diff1] [Diff2] [Errata]

INFORMATIONAL
Errata Exist

Internet Engineering Task Force (IETF)
Request for Comments: 6819
Category: Informational
ISSN: 2070-1721

T. Lodderstedt, Ed.
Deutsche Telekom AG
M. McGloin
IBM
P. Hunt
Oracle Corporation
January 2013

OAuth 2.0 Threat Model and Security Considerations

[Docs] [txt|pdf|xml|html] [Tracker] [WG] [Email] [Diff1] [Diff2] [Nits]

Versions: (draft-lodderstedt-oauth-security-topics)
00 01 02 03 04 05 06 07 08 09 10 11
12 13 14

urity considerations for OAuth,
ification, based on a comprehensive
tocol.

Web Authorization Protocol
Internet-Draft
Intended status: Best Current Practice
Expires: August 13, 2020

T. Lodderstedt
yes.com
J. Bradley
Yubico
A. Labunets

D. Fett
yes.com
February 10, 2020

tandards Track specification; it is
es.

OAuth 2.0 Security Best Current Practice
draft-ietf-oauth-security-topics-14

@aaronpk

STOLEN API KEYS

TWITTER

@aaronpk

https://www.theregister.co.uk/2013/03/08/twitter_oauth_leaked_keys/

Log in | Sign up | Forums

Serverless | M³ | CLL | Events | Whitepapers | The Next Platform

The Register®

Biting the hand that feeds IT

DATA CENTRE SOFTWARE SECURITY DEVOPS BUSINESS PERSONAL TECH SCIENCE EMERGENT TECH BOOTNOTES LECTURES

Data Centre ▸ Networks

Leaked: The 'secret OAuth app keys' to Twitter's VIP lounge

Rogue apps could pose as micro-blogging s
Very Important Programs

By John Leyden 8 Mar 2013 at 15:03 13

Twitter's private OAuth login keys, used by the website's offici
applications to get preferential treatment from the micro-blogg
have apparently been leaked. The secret credentials could no
software to masquerade as an approved Twitter client.

A set of key pairs uploaded to Github are supposedly used by
iPhone, Android, iPad, Mac OS X and Windows Phone, Twitte
Google TV, and TweetDeck. Once authenticated, these progra
access to features unavailable to clients that don't have Twitte
approval. But they need to store the keys somewhere on the u
computer or gadget, and it appears someone has found them.

The keys were first posted on the GitHub website five months
were updated within the past day; GitHubbers claim the data i

So now unapproved apps could send this information to Twitte

https://alexbilbie.com/2013/03/twitter-official-clients-have-their-oauth-keys-leak/

Twitter's official clients have their OAuth keys leak

07 Mar 2013

Someone has posted a Github Gist with all of the client identifiers and se
on various platforms.

ritive that you don't screw up like Faceb
direct URIs.

bsolute and stored on the authorisation

about this?

r all of their clients with new keys and
that haven't been updated will sudden
used and frustrated. Also it will be trivia

with some sort of mechanism to allow t
eys on the fly but then these could be
e attack.

Threatpost, Inc [US] | https://threatpost.com/twitter-oauth-api-keys-leaked-030713/77597/

threatpost

Cloud Security / Malware / Vulnerabilities / Privacy / HackerOne Spotlight

Twitter OAuth API Keys Leaked

Author:
Michael Mimoso

March 7, 2013 / 4:51 pm

2 minute read

Share this article:

The OAuth keys and secrets that official Twitter applications use to access

2013

@aaronpk

re4k / 3878505.md

Created 5 years ago

key.md

Twitter Official Consumer Key

Twitter for iPhone

```
Consumer key: IQKbtAYlXLripLGPWd0HUA
Consumer secret: GgDYlkSvaPxGxC4X8liwpUoqKwwr3lCADbz8A7ADU
```

Twitter for Android

```
Consumer key: 3nVuSoBZnx6U4vzUxf5w
Consumer secret: Bcs59EFbbsdF6Sl9Ng71smgStWEGwXXKSjYvPVt7qys
```

Twitter for Android Sign-Up

```
Consumer key: RwYLhxGZpMqsWZENFVw
Consumer secret: Jk80YVGqc7Iz1IDEjCI6x3ExMSBnGjzBAH6qHcWJlo
```

ANYONE CAN IMPERSONATE THE TWITTER APPS

DON'T PUT SECRETS IN NATIVE APPS!

https://developer.okta.com/blog/2019/01/22/oauth-api-keys-arent-safe-in-mobile-apps

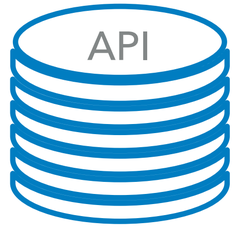
PKCE

(pronounced "pixie")

PROOF-KEY FOR CODE EXCHANGE

RFC 7636

@aaronpk



User: I'd like to use this great app

App: Hang on while I generate a new secret and hash it

App: Please go to the authorization server to grant me access, **take this hash with you**

User: I'd like to log in to this app, **here's the hash**

AS: Here is a temporary code the app can use

User: Here is the temporary code, please use this to get a token

App: Here's the code, **and the plaintext secret**, please give me a token

AS: **Let me verify the hash of that secret**... ok here is an access token!

App: Please let me access this user's data with this access token!

AppAuth.io

iOS / Android / JavaScript

@aaronpk

photo by flickr.com/quidox

ALG=NONE

JWT

@aaronpk

Critical vulnerabilities in JSON Web Token libraries

Which libraries are vulnerable to attacks and how to prevent them.

Tim McLean March 31, 2015

Subscribe to more awesome content!

tl;dr If you are using n
jwt or jsjwt with asym
ES384, ES512) please u
more information on t

This is a guest post fro

npm Search packages Search log in

severity critical

Verification Bypass

jsonwebtoken

Advisory Versions

Advisory timeline

Reported
Oct 17th, 2015

Published
Advisory publish
Apr 1st, 2015

NIST NVD MENU

Information Technology Laboratory

NATIONAL VULNERABILITY DATABASE

VULNERABILITIES

CVE-2015-9235 Detail

Current Description

In jsonwebtoken node module before 4.2.2 it is possible for an attacker to bypass verification when a token digitally
asymmetric key (RS/ES family) of algorithms but instead the attacker send a token digitally signed with a symmetri

Source: MITRE
Description Last Modified: 05/30/2018
View Analysis Description

Impact

CVSS v3.0 Severity and Metrics:

CVSS v2.0 Severity and Metrics:

Sjoerd Langkemper

Web application security

Attacking JWT authentication

Sep 28, 2016

JSON Web Tokens or JWTs are used by some web applications instead of traditional session cookies. Because of their statelessness and the signature implementation there are some security issues that are specific to JWTs. This post describes some ways you can verify that a JWT implementation is secure.

About JWTs

What is a JWT

A JWT (JSON Web Token) is a string that contains a signed data structure, typically used to

2015

JWTS ARE OFTEN USED
FOR API AUTHENTICATION
AND AS OAUTH ACCESS TOKENS

An Example JWT


```
eyJraWQiOiJvQ1JjR3RxVDhRV2tJR0MyVXpmcEZUczVqSkdnM00zSTNOMHgtZDJhSFNNIiwiYW
xnIjoiUlMyNTYifQ.eyJ2ZXIiOjEsImp0aSI6IkFULkp3eVRTcTlqNDU0bDNTNmRTM1VTV1hMV
VpwekdKdWNSd1ZEbFZCNWNIc3cuVVM1V1NGYVFiQllUMC9GM2tjMG8vK1ZUY3VZZzdwVnZqZXZ
TT3hkUHhCMD0iLCJpc3MiOiJodHRwczovL2Rldi0zOTYzNDMub2t0YXByZXZpZXcuY29tL29hd
XRoMi9kZWZhdWx0IiwiYXVkIjoiYXBpOi8vZGVmYXVsdCIsImlhdCI6MTU0MzgwMzAyNSwiZXh
wIjoxNTQzODA2NjI1LCJjaWQiOiIwb2FoenBwM3RjcEZyZmNXSTBoNyIsInVpZCI6IjAwdWkwZ
mpraWV5TDQ2bWEwMGg3Iiwic2NwIjpbIm9mZmxpbmVfYWNjZXNzIiwicGhvdG8iXSwic3ViIjo
iaW5xdWlzaXRpdmUtYWxiYXRyb3NzQGV4YW1wbGUuY29tIn0.ncVkzcc6qrFJSXE3-5UsRu_kH
vbwIMKYL3PFaMwReYTquPAcOQ8t93xF0bxbS8wrP0udCDvk6eYq4VbjoFdD59Yy6ltz0OKQl3-
g8uFg2RwqTBMOKR0mYtQH0RCr9ORhSsmKolaDDt4TcRX78ZOAyhZ_Qg_UcEoHM4uZikpzBJYpY
KbCCfbx-6FzYyHuvevSFzURISYpSHv3nbzirkEzKbOv7eZlg1cCYBdUoGuVBskyHxfMxFpoKQU
3mwIFdlQJR8LZ8hA_5ZdYjjMeSXfjnhlP2rppJiHy1NreGXXcUsUA74V2t_keY44deTrnPgoFO
Se9IchWqcj6sDMDutC4ag
```

ID Token: JWT


```
eyJraWQiOiJiRmxZbmkzLXRhMXFSa0lFellHc2tLeFFRVUJvczZnOU9RQnRmNm9xcUxJIiwiYWxnI
joiUlMyNTYifQ

.

eyJzdWIiOiIwMHVjcTNid2o0V25JcTNnejBoNyIsIm5hbWUiOiJQYWRtYS0yIEdvdmluZGFyYWphb
HUiLCJsb2NhbGUiOiJlbi1VUyIsInZlciI6MSwiaXNzIjoiaHR0cHM6Ly9wYWRtYWdvdmluZGFyYW
phbHUub2t0YXByZXZpZXcuY29tL29hdXRoMi9kZWZhdWx0IiwiYXVkIjoiMG9hZDlydTd0endmNUF
qcGIwaDcgIiwiaWF0IjoxNTI0NTk0OTEwLCJleHAiOjE1MjQ1OTg1MTAsImp0aSI6IklELklfNUc4
RzhWdXowMHJvYl9aSzlja3J0T0pseVdwNzhxMU5naGV2QlJ6dkEiLCJhbXIiOlsicHdkIl0sImlkc
CI6IjAwb2NxM2J3aTFoTnpRT3B5MGg3IiwibW9uY2UiOiJhYmMiLCJwcmVmZXJyZWRfdXNlcm5hbW
UiOiJwYWRtYS5nb3ZpbmRhcmFqYWx1QG9rdGEuY29tIiwiZ2l2ZW5fbmFtZSI6IlBhZG1hIiwibWl
kZGxlX25hbWUiOiJLcmlzaG5hIiwiZmFtaWx5X25hbWUiOiJHb3ZpbmRhcmFqYWx1Iiwiem9uZWlu
Zm8iOiJBbWVyaWNhL0xvc19BbmdlbGVzIiwidXBkYXRlZF9hdCI6MTUyNDU5NDM2MSwiYXV0aF90a
W1lIjoxNTI0NTk0OTA3fQ

.

HvMYW8XbdCf1BW-
ZfHQ1odaAYJjZqKkh1NUkHW0clk6J7pYunn8jllbIp0IhSjcCn6PBIlZPrrE0dkuyjvdHjVI8ALQN
wtM7FnIs9H6gCH0oONx4EL4K-Ef4d_w46qeqsCwMClvNoaE3c2I5-kON-
uJUlaefbnr6Al_y9z5mvLyDynf9IjrOyTPoIrgk9V46l28Aulp4dJhqBtZfpYyVbKrXawHSO5FvKT
DMPBhQgxt0_6PKG7sSkhbMeBicIc35SJJaXt81KSfkYDUp5s1UQ74ATHrtLe7HMU1yp_KajgYUKxM
XO5NiXpeNEHzarAOWzLHblrQcgkpuJbY3KM1HHg
```

Attacking a JWT

header

```
{
 "typ": "JWT",
 "alg": "RS256"
}
```

claims

```
{
 "ver": 1,
 "jti": "AT.JwyTSq9j454l3S6dS3USWXLUZpzGJucRwVDlVB5cHsw.US5WSFaQbBYT0/F3kc0o/+VTcuYg7pVvjevSOxdPxB0=",
 "iss": "https://dev-396343.oktapreview.com/oauth2/default",
 "aud": "api://default",
 "iat": 1543803025,
 "exp": 1543806625,
 "cid": "0oahzpp3tcpFrfcWI0h7",
 "uid": "00ui0fjkieyL46ma00h7",
 "scp": [
  "offline_access",
  "photo"
 ],
 "sub": "inquisitive-albatross@example.com"
}
```

signature

Attacking a JWT

header

```
{
 "typ": "JWT",
 "alg": "none"
}
```

claims

```
{
 "ver": 1,
 "jti": "AT.JwyTSq9j454l3S6dS3USWXLUZpzGJucRwVDlVB5cHsw.US5WSFaQbBYT0/F3kc0o/+VTcuYg7pVvjevSOxdPxB0=",
 "iss": "https://dev-396343.oktapreview.com/oauth2/default",
 "aud": "api://default",
 "iat": 1543803025,
 "exp": 1543806625,
 "cid": "0oahzpp3tcpFrfcWI0h7",
 "uid": "00ui0fjkieyL46ma00h7",
 "scp": [
  "offline_access",
  "photo"
 ],
 "sub": "inquisitive-albatross@example.com"
}
```

Treat the JWT header as
untrusted external information

Never let the JWT header
determine your verification mechanism

Thankfully most JWT libraries
fixed this in 2015-2016

OAUTH PHISHING

GOOGLE

@aaronpk

BBC undefined News Sport Weather Shop Earth Travel More Search

NEWS

Find local news

Home UK World Business Politics Tech Science Health Education Entertainment & Arts Video & Audio More

Technology

Google Docs phishing email 'cost Minnesota $90,000'

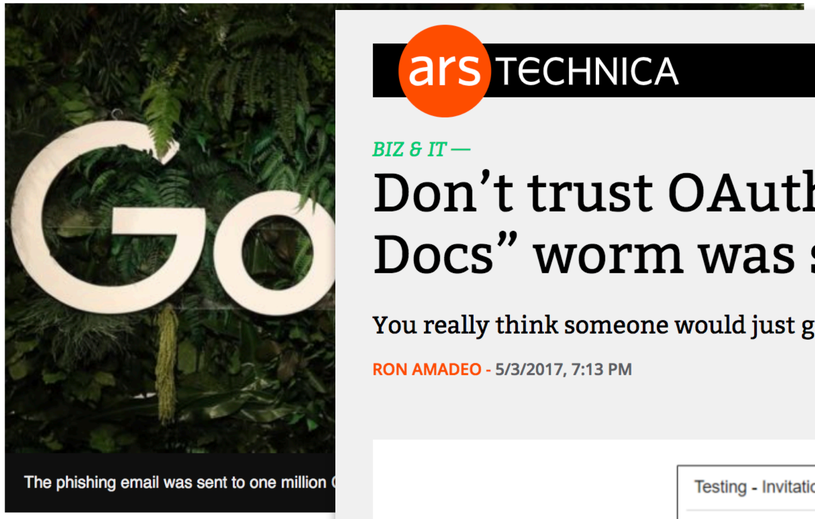
8 May 2017 | Technology

Share

Top Stories

Tough task ahead as Macron basks in win

Emmanuel Macron takes his first steps as France's president-elect in establishing his team after beating Marine Le Pen.

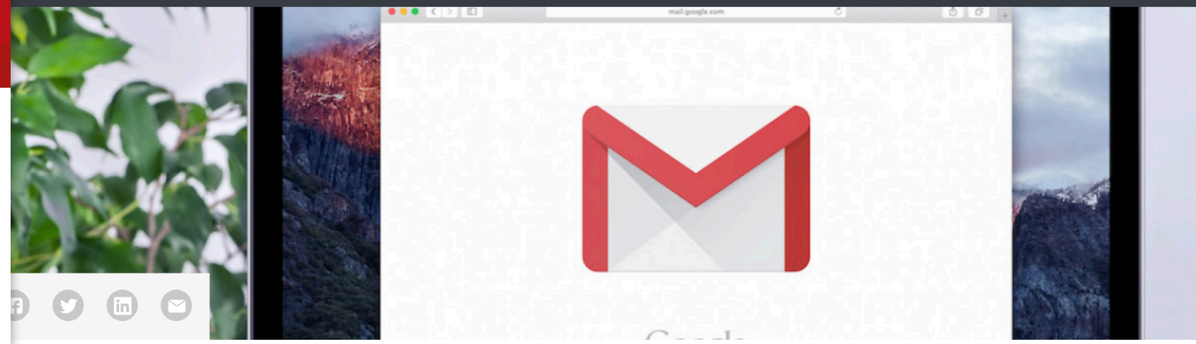


The phishing email was sent to one million

A phishing email that targeted Gma of Minnesota $90,000 (£69,400).

About 2,500 state employees received information security officer.

Blog Home Categories Archive Keep In Touch



ars TECHNICA BIZ & IT TECH SCIENCE POLICY CARS GAMING & CULTURE STORE

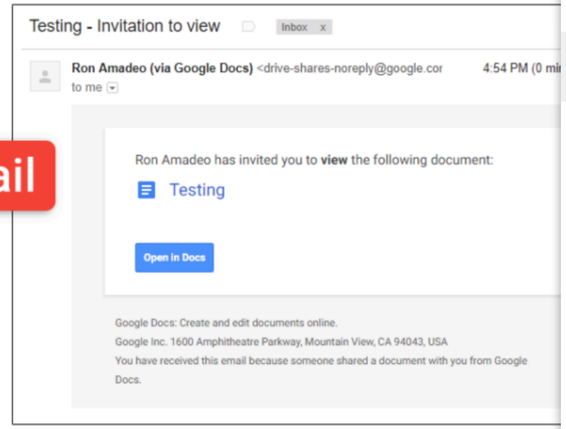
BIZ & IT —

Don't trust OAuth: Why the "Google Docs" worm was so convincing

You really think someone would just go on the Internet and tell lies?

RON AMADEO - 5/3/2017, 7:13 PM

Real E-mail



DUO LABS

MAY 4TH, 2017

Gmail OAuth Phishing Goes Viral

On Wednesday, a Gmail phishing attack leveraging OAuth spread quickly to multiple users. There are a number of features that you need to be aware of that made this attack incredibly successful, as well as ways to protect yourself or your employees that are detailed below.

Google Docs would like to

The State of Security

NEWS. TRENDS. INSIGHTS.

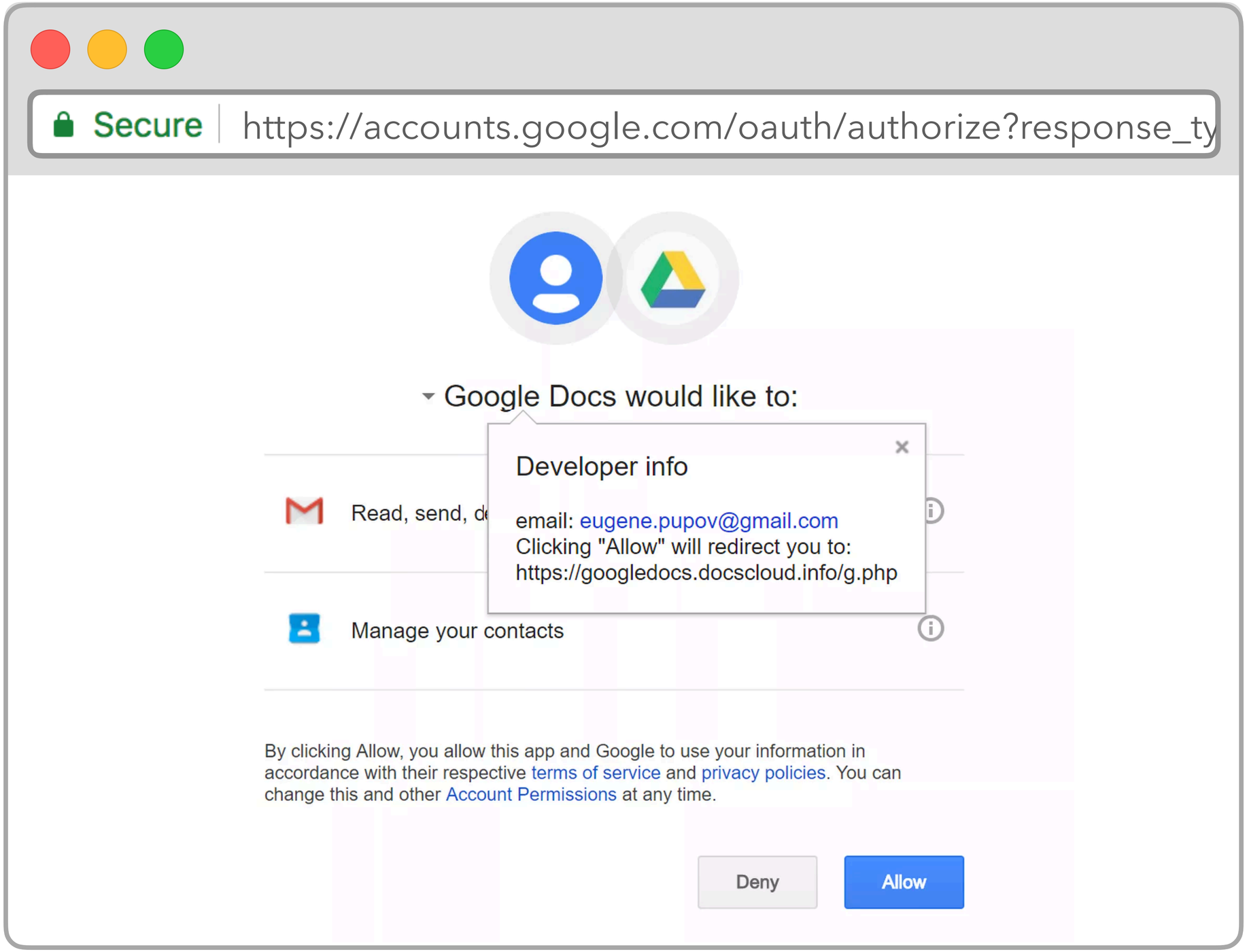
FEATURED ARTICLES TOPICS VERT RESOURCES

"Google Docs" Worm Ransacks Gmail Users' Contact Lists – What You Need to Know

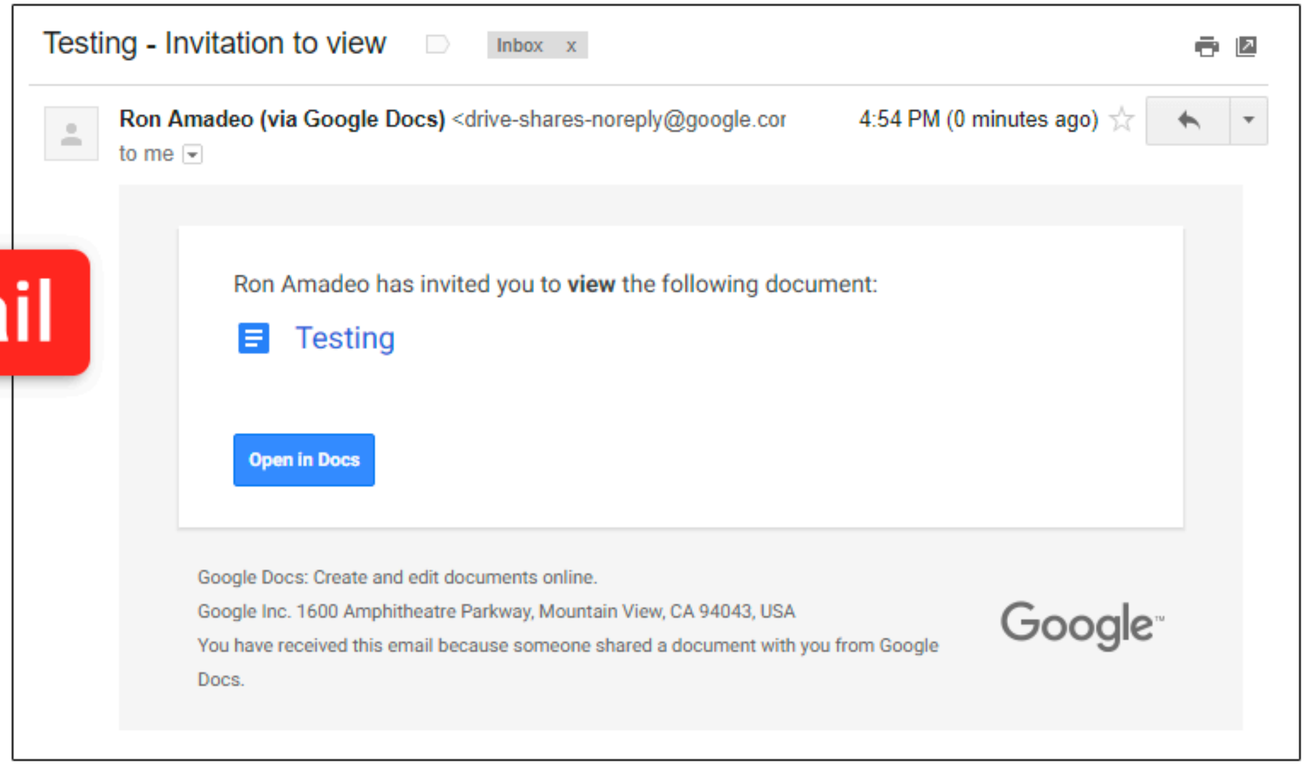
GRAHAM CLULEY Follow @gcluley

MAY 4, 2017 | IT SECURITY AND DATA PROTECTION

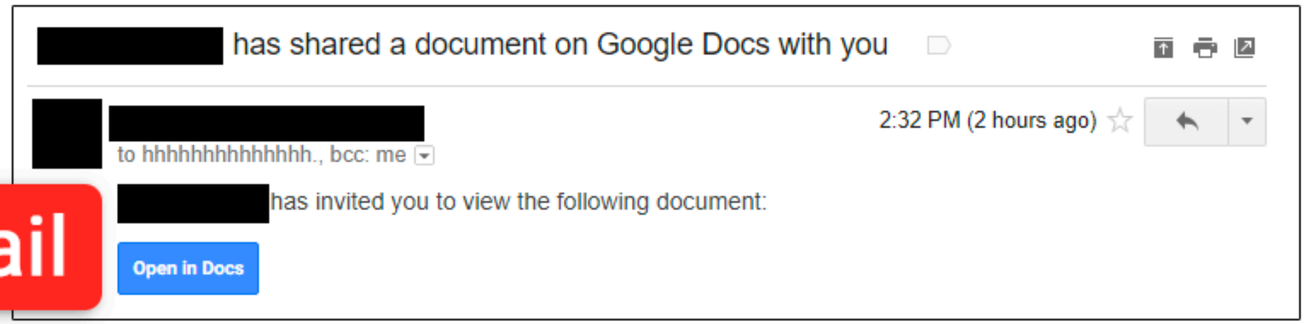
2017


Secure
https://accounts.google.com/oauth/authorize?response_ty
Google Docs would like to:
Developer info
email: eugene.pupov@gmail.com
Clicking "Allow" will redirect you to:
https://googledocs.docscloud.info/g.php
Read, send, d
Manage your contacts
By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.
Deny
Allow

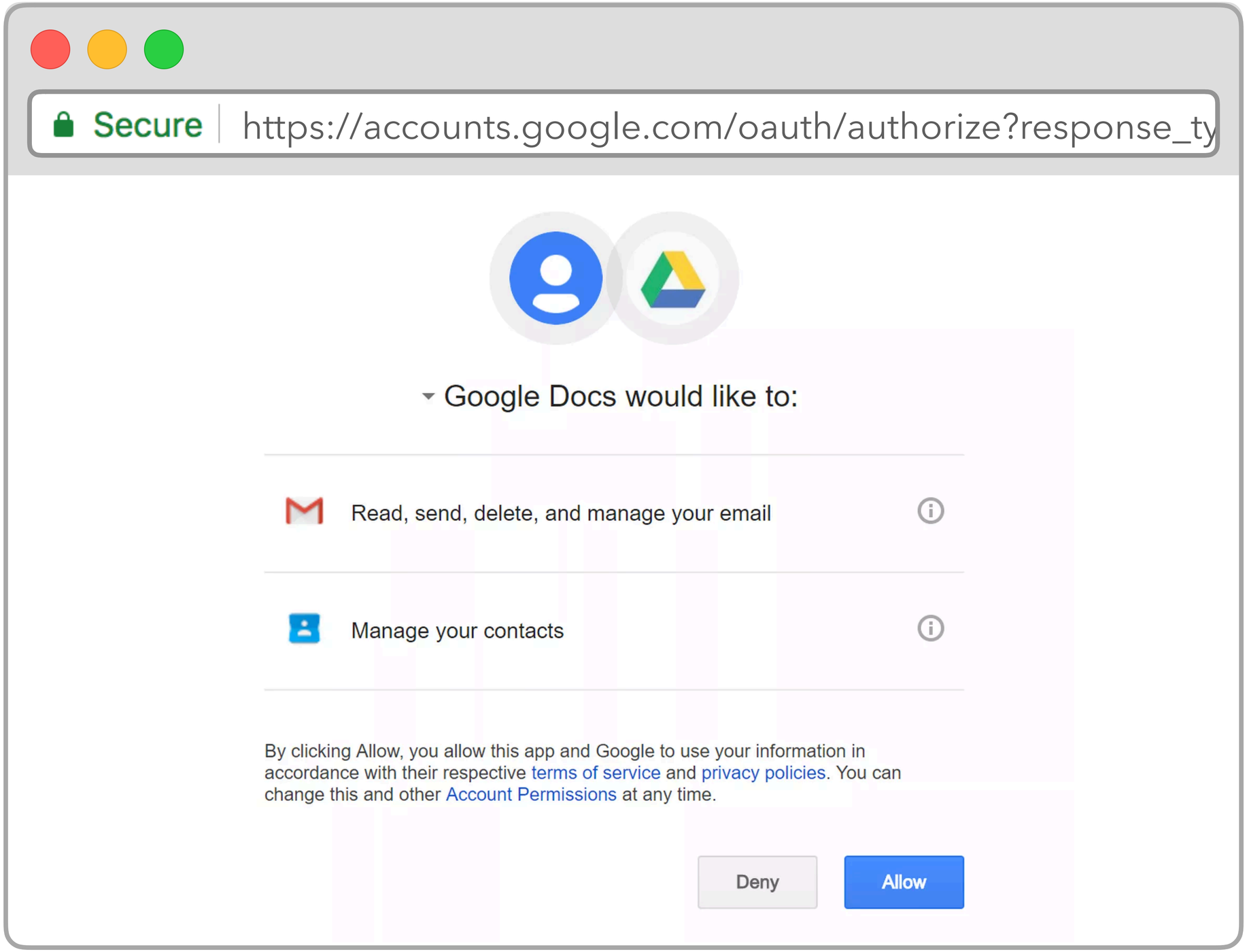
Real E-mail



Fake E-mail



https://arstechnica.com/information-technology/2017/05/dont-trust-oauth-why-the-google-docs-worm-was-so-convincing/


Secure
https://accounts.google.com/oauth/authorize?response_ty
Google Docs would like to:
Read, send, delete, and manage your email
Manage your contacts
By clicking Allow, you allow this app and Google to use your information in accordance with their respective terms of service and privacy policies. You can change this and other Account Permissions at any time.
Deny
Allow


Gmail
@gmail
Follow
We are investigating a phishing email that appears as Google Docs. We encourage you to not click through, & report as phishing within Gmail.
1:15 PM - 3 May 2017
8,511 Retweets 4,159 Likes
112
8.5K
4.2K

New Google Docs phishing scam, almost undetectable

Update: scam banned | /r/all

I received a phishing email today, and very nearly fell for it. I'll go through the steps here:

1. I received an email that a Google Doc had been shared with me. Looked reasonably legit, and I recognized the sender.
2. The button's URL was somewhat suspicious, but still reasonably Google based.
3. I then got taken to a *real* Google account selection screen. It already knew about my 4 accounts, so it's really signing me into Google.
4. Upon selecting an account, no password was needed, I just needed to allow "Google Docs" to access my account.
5. If I click "Google Docs", it shows me it's *actually* published by a random gmail account, so that user would receive full access to my emails (and could presumably therefore perform password resets etc).
6. Shortly afterwards I received [a followup real email from my contact] (https://i.imgur.com/DGSyy1L.png), informing me: "Delete this is a spam email that spreads to your contacts."

To summarise, this spam email:

- Uses the existing Google login system
- Uses the name "Google Docs"
- Is only detectable as fake if you happen to click "Google Docs" whilst granting permission
- Replicates itself by sending itself to all your contacts
- Bypasses any 2 factor authentication / login alerts
- Will send scam emails to everyone you have ever emailed

the_mighty_skeetadon **Verified Google dude** 5.8k points · 2 years ago · *edited 2 years ago* 2

Googler here -- I'm escalating to the correct engineering and product teams now.

Edit: This is now resolved. Less than a half-hour after escalation, wow! =). Here's the official Google statement:

> We have taken action to protect users against an email impersonating Google Docs, and have disabled offending accounts," the company said in a statement. "We've removed the fake pages, pushed updates through Safe Browsing, and our abuse team is working to prevent this kind of spoofing from happening again. We encourage users to report phishing emails in Gmail.

Share Report Save

the_mighty_skeetadon **Verified Google dude** 1.7k points · 2 years ago · *edited 2 years ago*

Official response from the eng manager in charge of this stuff: "yes, I am on it" =). I'd bet it will be fixed and fully rolled out in a few hours or less.

Final edit: problem is resolved. I clicked the link and got an "oauth client disabled" message. Not pretty, but at least you won't get phished.

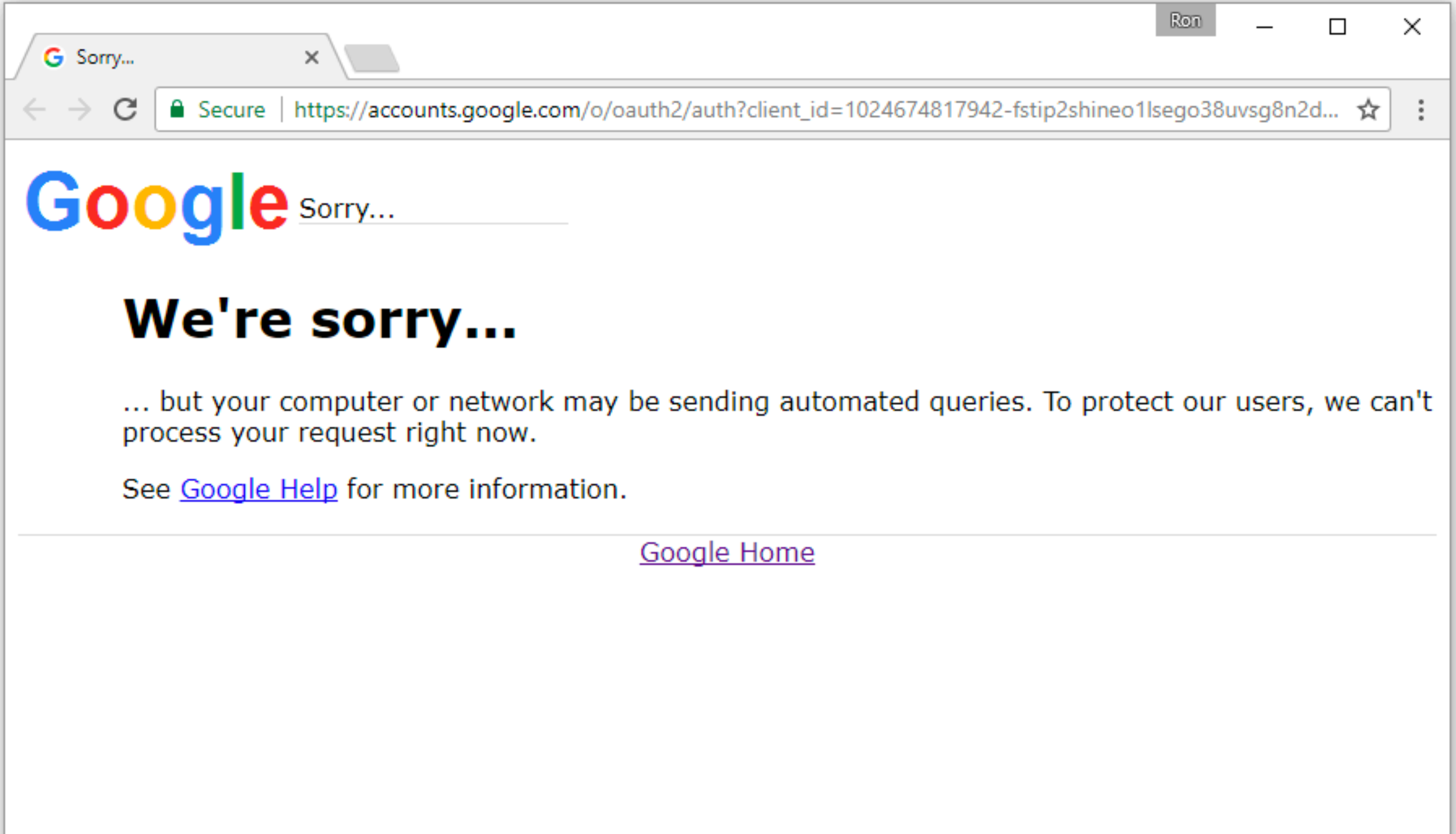
Share Report Save

[deleted] 725 points · 2 years ago

This is such an impressive turnaround time for a problem, but I'm not surprised at all that Google can pull off such a quick fix. Bravo.

Share Report Save

VIEW ENTIRE DISCUSSION (1.1K COMMENTS)


Ron
Sorry...
Secure
https://accounts.google.com/o/oauth2/auth?client_id=1024674817942-fstip2shineo1lsego38uvsg8n2d...
Google
Sorry...
We're sorry...
... but your computer or network may be sending automated queries. To protect our users, we can't process your request right now.
See Google Help for more information.
Google Home

improperly issued

~~STOLEN~~ ACCESS TOKENS

FACEBOOK

@aaronpk

2018

WIRED

The Facebook Hack Exposes an Internet-Wide Failure

SHARE

SHARE

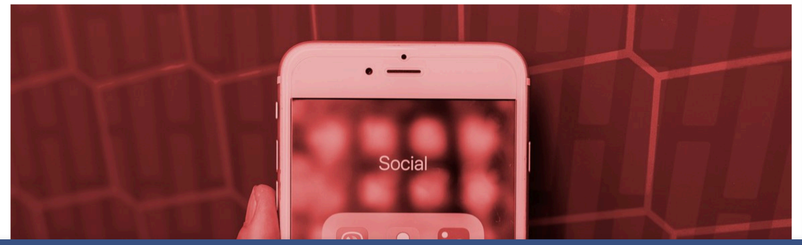
TWEET

COMMENT

EMAIL

ISSIE LAPOWSKY SECURITY 10.02.18 10:12 AM

THE FACEBOOK HACK EXPOSES AN INTERNET-WIDE FAILURE



MOTHERBOARD Moveable Hacking Environment Space Gaming Health Tech Science Influence

FACEBOOK | By Lorenzo Franceschi-Bicchierai and Jason Koebler | Sep 28 2018, 2:30pm

How 50 Million Facebook User Were Hacked

Facebook revealed more details about how hackers exploited three distinct bugs to get the ability to control up to 50 million users' accounts.

facebook Newsroom

Like 522K Share Search in Newsroom

Home News Company Info Directory Media Gallery Inside Feed Public Policy Investor Relations

uth0 Platform Solutions Why Auth0 Developers Pricing Talk to Sales Log In SIGN UP

BLOG

Early Look at Facebook Access Token Security Breach

Almost 90 millions Facebook users were affected by a breach that compromised access tokens to the platform.

September 28, 2018

Security Update

Contact Us

press@fb.com

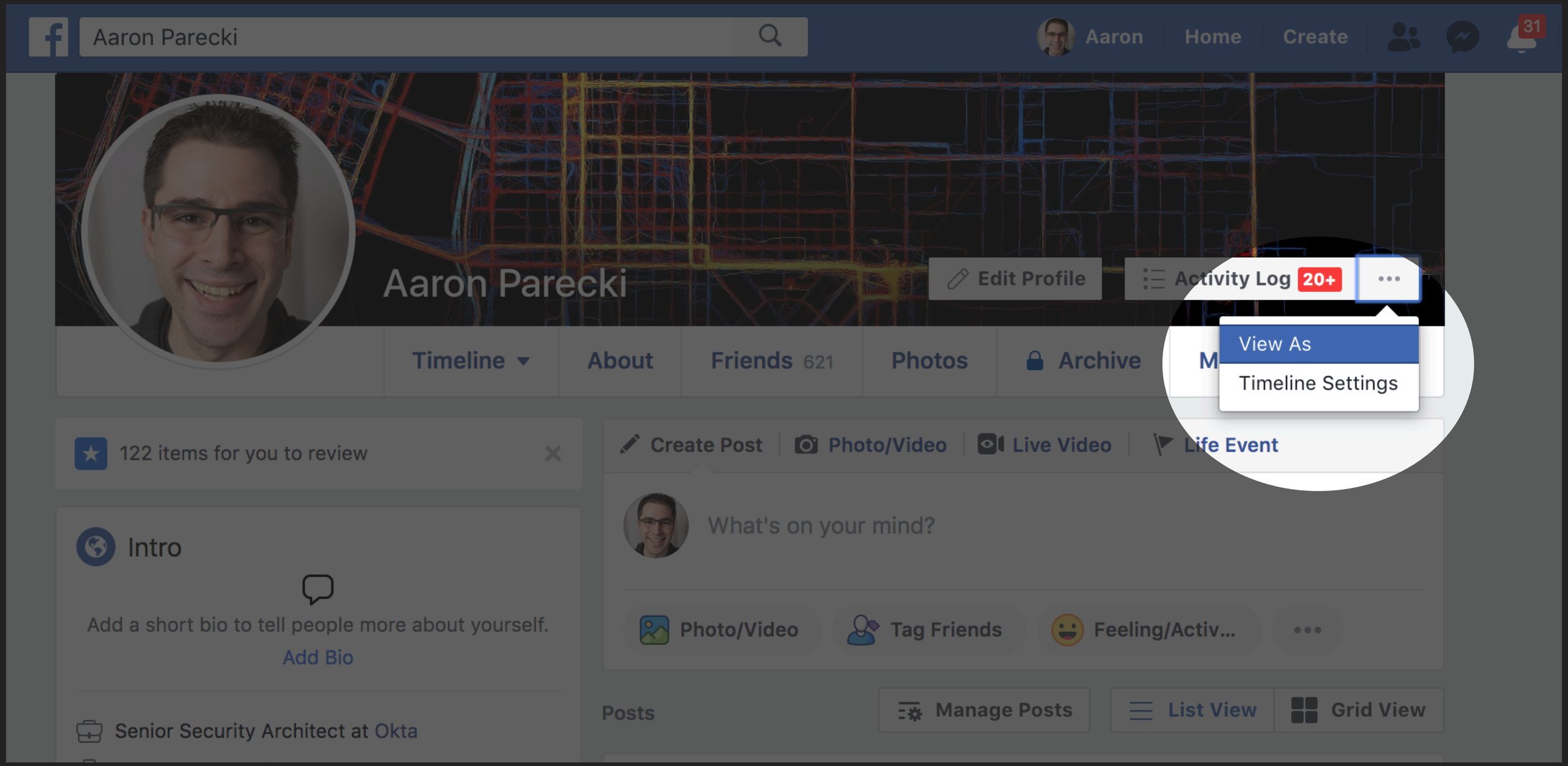
Categories

Company News
Product News
Community Boost
Data & Privacy
Innovation
Integrity & Security
Safety & Well-Being
Social Impact

"The vulnerability was the result of the interaction of three distinct bugs"

- Guy Rosen, VP of Product Management, Facebook

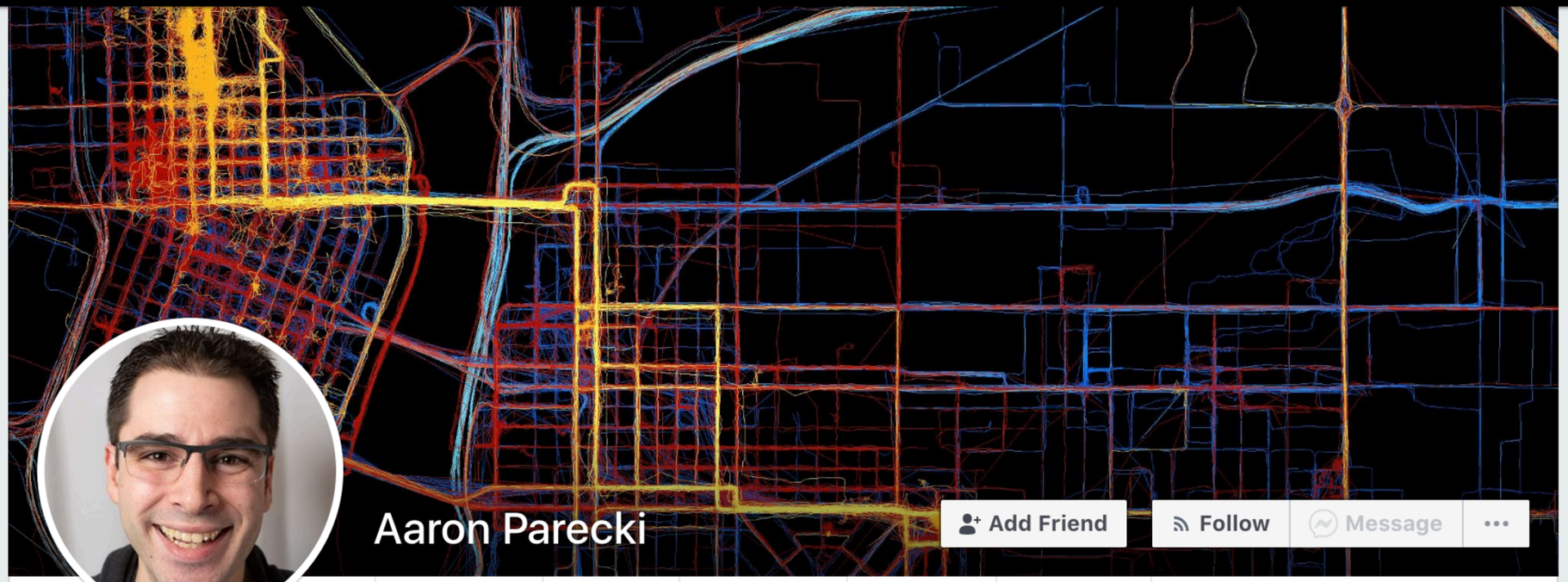
https://newsroom.fb.com/news/2018/09/security-update/


Aaron Parecki
Aaron
Home
Create
31
Aaron Parecki
Edit Profile
Activity Log 20+
View As
Timeline Settings
Timeline
About
Friends 621
Photos
Archive
122 items for you to review
Create Post
Photo/Video
Live Video
Life Event
What's on your mind?
Intro
Add a short bio to tell people more about yourself.
Add Bio
Photo/Video
Tag Friends
Feeling/Activ...
Senior Security Architect at Okta
Posts
Manage Posts
List View
Grid View

Aaron Parecki

Aaron | Home | Create | 31

This is what your profile looks like to: Public



Aaron Parecki

Add Friend | Follow | Message | ...

Timeline | About | Friends | Photos | More

DO YOU KNOW AARON?

To see what he shares with friends, send him a friend request.

Add Friend

Intro

- Senior Security Architect at Okta
- Former CTO, Esri R&D Center, Portland at Esri

Aaron Parecki
June 4

If you are curious about "Sign In with Apple," I walk through exactly how it works and what it looks like in this post.

https://developer.okta.com/.../what-the-heck-is-sign-in-with-...

The vulnerability was the result of the interaction of three distinct bugs:

First: View As is a privacy feature that lets people see what their own profile looks like to someone else.

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

First: View As is a privacy feature that lets people see what their own profile looks like to someone else. View As should be a view-only interface.

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

First: View As is a privacy feature that lets people see what their own profile looks like to someone else. View As should be a view-only interface. However, for one type of composer (the box that lets you post content to Facebook) — specifically the version that enables people to wish their friends happy birthday —

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

First: View As is a privacy feature that lets people see what their own profile looks like to someone else. View As should be a view-only interface. However, for one type of composer (the box that lets you post content to Facebook) — specifically the version that enables people to wish their friends happy birthday — View As incorrectly provided the opportunity to post a video.

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

Second: A new version of our video uploader (the interface that would be presented as a result of the first bug), introduced in July 2017,

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

Second: A new version of our video uploader (the interface that would be presented as a result of the first bug), introduced in July 2017, incorrectly generated an access token that had the permissions of the Facebook mobile app.

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

Third: When the video uploader appeared as part of View As,

https://newsroom.fb.com/news/2018/09/security-update/

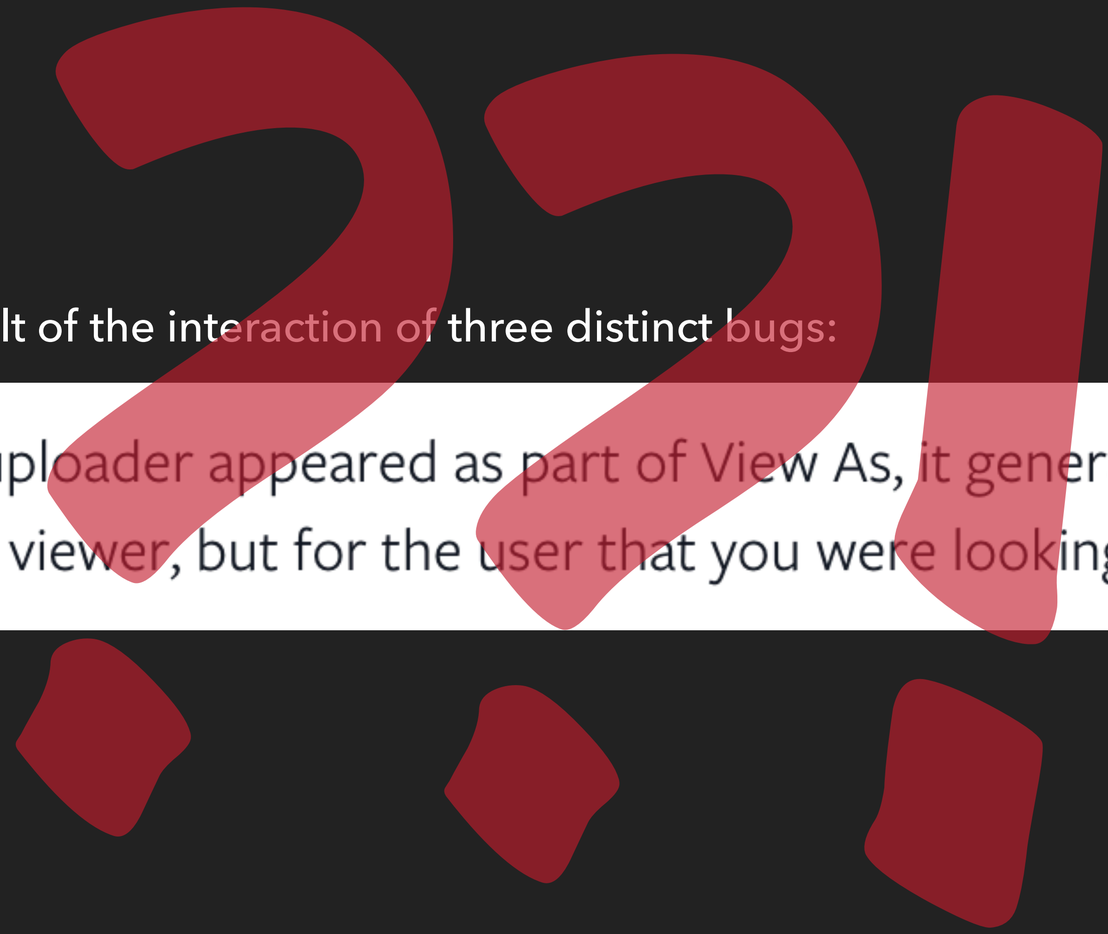
The vulnerability was the result of the interaction of three distinct bugs:

Third: When the video uploader appeared as part of View As, it generated the access token not for you as the viewer,

https://newsroom.fb.com/news/2018/09/security-update/

The vulnerability was the result of the interaction of three distinct bugs:

Third: When the video uploader appeared as part of View As, it generated the access token not for you as the viewer, but for the user that you were looking up.



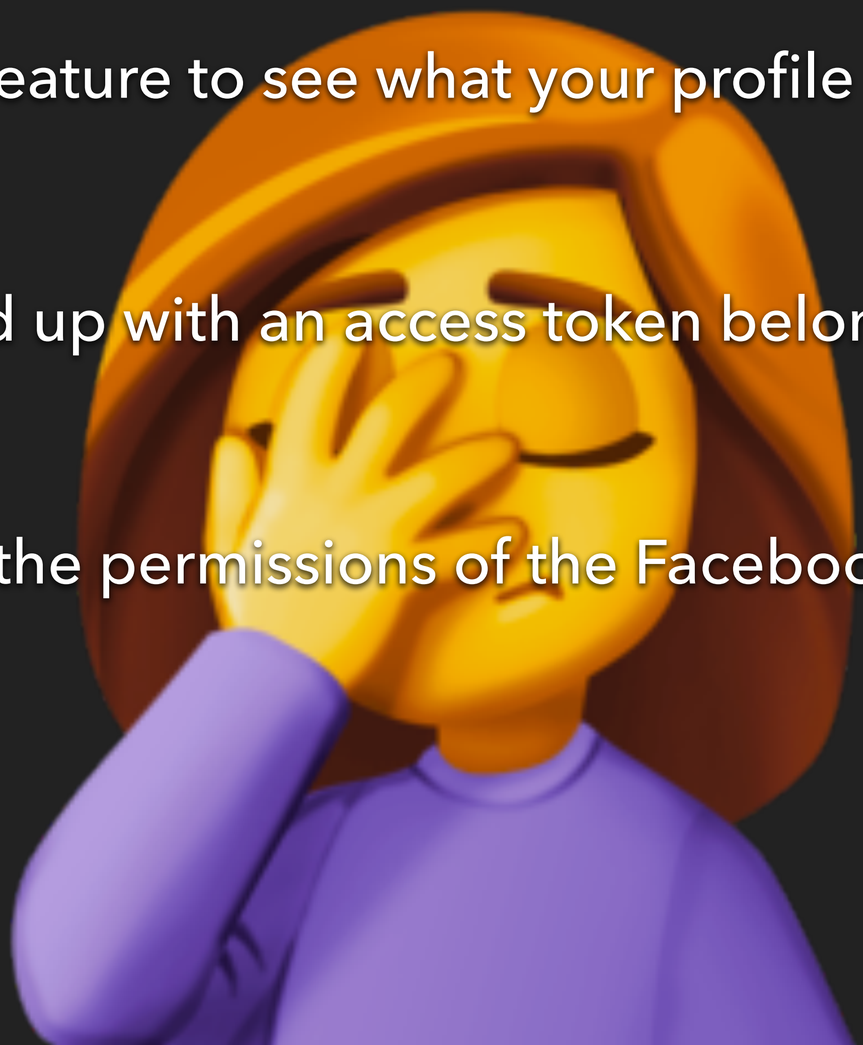
https://newsroom.fb.com/news/2018/09/security-update/

@aaronpk

By using the "View As" feature to see what your profile looks like to someone else,

you would end up with an access token belonging to that user,

which had the permissions of the Facebook mobile app.



Keep clean security boundaries

even for internal applications

Don't let applications pretend
to be other applications or other users

Thank You!

@aaronpk

aaronpk.com

oauth.wtf