

#### The following work is all under NDA

Carolyn Huynh careOhlyn.github.com

# **Self Driving Cloud**

IAM Role Recommender



#### Carolyn Huynh | Google Cloud Security



#### Lead UX Designer

#### Role

I was the first designer to join this team and ever since, I've been the Lead UX Designer and thought leader behind the suite of self driving cloud/policy intelligence tools for three years now.

I've lead every product within SDC/PI from inception to launch cycles. Since I was the first to join the team, I've been the go-to designer for all x-functional teams within the Self Driving Cloud ecosystem.

I've also been responsible for all leading the initial vision work concepts as well.

# Applying Machine Learning Intelligence and Analytics to security products

#### What are we building?

IAM Role Recommender is the first product launch within the Self Driving Cloud suite of products. Role Recommender is built on top of IAM (Identity & Access Management) policy.

At the time, no other competitors (AWS, Azure) had this tooling available and we were moving at lightning speed to ship this product out tomorrow.



#### Who are we building it for?

Security teams for our existing enterprise customers that are on Google Cloud Platform. Companies such as Spotify, Target and Snapchat were some of the big names we worked with on this feature.

Also for security consultants (PWC etc) and 3rd party products that might want to take advantage of this functionality.



# I need help adhering to the principle of least privilege.

Nicole
Security Engineer at {Tech Company}



#### **The Problem Statement:**

Least Privilege is hard to measure. It's also hard to get right.

#### More constraints

Constraints stacked on constraints stacked on constraints



### Understanding least privilege

There is no easy way to understand how well customers are doing in maintaining least privilege.

### Tracking least privilege

No easy way to track all the resources being created in an organization and who has access to them in the company.

### Revealing when it's over privileged

There is no easy way to see what permissions an individual is using

#### When to timebox access?

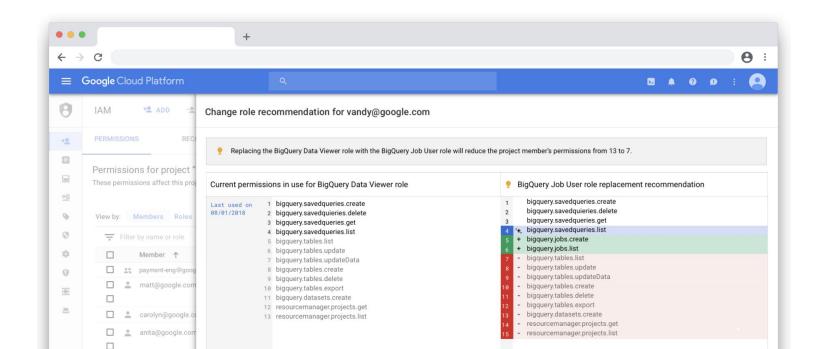
Most access tends to be indefinite

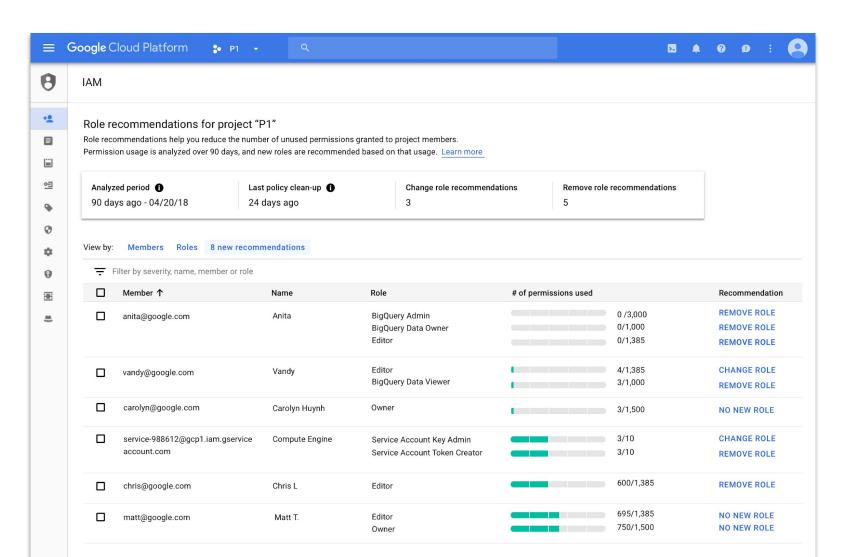
# The Big UX ask:

How does one visualize surfacing machine learning recommendations while also building confidence in users so that they apply them?

#### **Early Design Explorations**

#### 60+ iterations





	Google C	loud Platform	Q			>-		8	ø	:	9	
θ	IAM											
+2	Role reco	Role recommendations for project "P1"  Role recommendations help you reduce the number of unused permissions granted to project members.  Permission usage is analyzed over 90 days, and new roles are recommended based on that usage. Learn more										
• • • • • • • • • • • • • • • • • • •	24 day	ys ago	Role removal recommendations 5									
	View by:     Members     Roles     Recommendations								Unused permissions			
=		anita@google.com	Anita	BigQuery Admin BigQuery Data Owner Editor	Remove role Remove role			5 <u></u>		100	%	
		vandy@google.com	Vandy	Editor BigQuery Data Viewer	Remove role					99%		
		carolyn@google.com service-988612@gcp1.iam.gservic	Carolyn Huynh	Owner  Service Account Key Admin	③ Replace role					98%		
		account.com	e compute Engine	Service Account Token Creator	Remove role					49%		
		chris@google.com	Chris L	Editor	Replace role					47%	,	
		matt@google.com	Matt T.	Editor Owner	0					55%	)	

Something was not right.

We weren't visualizing machine learning intelligence in a more powerful way.



1

Back to the drawing board

We needed more research

2

Ran customer research sessions

A lot of qualitative chats

3

# Customer whiteboarding sessions

...mixed with hands on brainstorming

4

# Designing on the spot in front of customers

A quick and dirty yes? no?

## The Light Bulb Moment

I was designing with roles. But customers wanted to go as granular as the permissions within the roles, which are hidden in the UI and only exposed via the API.

# Research taught us that when it came to changing permissions, we had to be as *explicit* as possible

#### We talked to a lot of customers. A lot.

Based off foundational research, customer whiteboarding sessions, and several low-low fi concepts, my UX researcher and I were able to discern that in order for us to gain trust from our customers to apply Google recommendations, we needed to be as granular as possible.

After combing through all the qualitative data, I couldn't get my mind off off the granularity when it comes to displaying trust.

#### Especially in the security space.

So, I decided to bring back the trusty code diff and give it a makeover.

#### What a standard code diff looks like

#### the -/+ pattern is key here

```
+- (RACSignal *)enqueueRequest:(NSURLRequest *)request fetchAllPages:(BOOL)fetchAllPages;
        95
 82
             // Enqueues a request to fetch information about the current user by accessing
 83
             // a path relative to the user object.
 84
        98
            @@ -241,11 +255,13 @@ - (id)initWithServer:(OCTServer *)server
               NSString *userAgent = self.class.userAgent;
241
      255
242
      256
               if (userAgent != nil) [self setDefaultHeader:@"User-Agent" value:userAgent];
243
244

    self.parameterEncoding = AFJSONParameterEncoding;

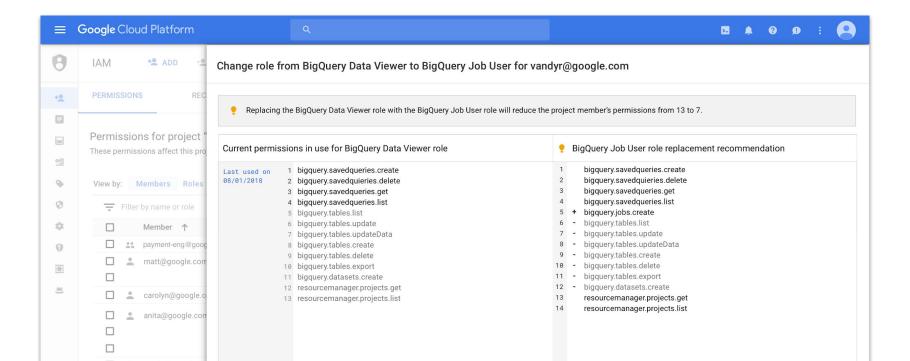
245
               [self setDefaultHeader:@"Accept" value:@"application/vnd.github.beta+json"];
246
247
      258
                [AFHTTPRequestOperation addAcceptableStatusCodes:[NSIndexSet indexSetWithIndex:OCTClientNotModifiedStatusCode]
248

    [AFJSONRequestOperation addAcceptableContentTypes:[NSSet setWithObject:@"application/vnd.github.beta+json"]];

      259
            + NSString *contentType = [NSString stringWithFormat: @ application/vnd.github. % @ + json", OCTClientAPIVersion];
             + [self setDefaultHeader:@"Accept" value:contentType];
      262
            + [AFJSONRequestOperation addAcceptableContentTypes:[NSSet setWithObject:contentType]];
      263
            + self.parameterEncoding = AFJSONParameterEncoding;
      264
249
      265
                [self registerHTTPOperationClass:AFJSONRequestOperation.class];
```

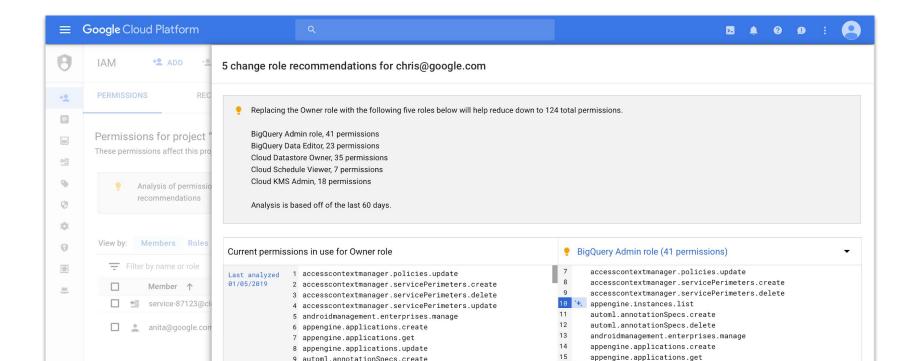
#### Wait. We're still missing the ML part.

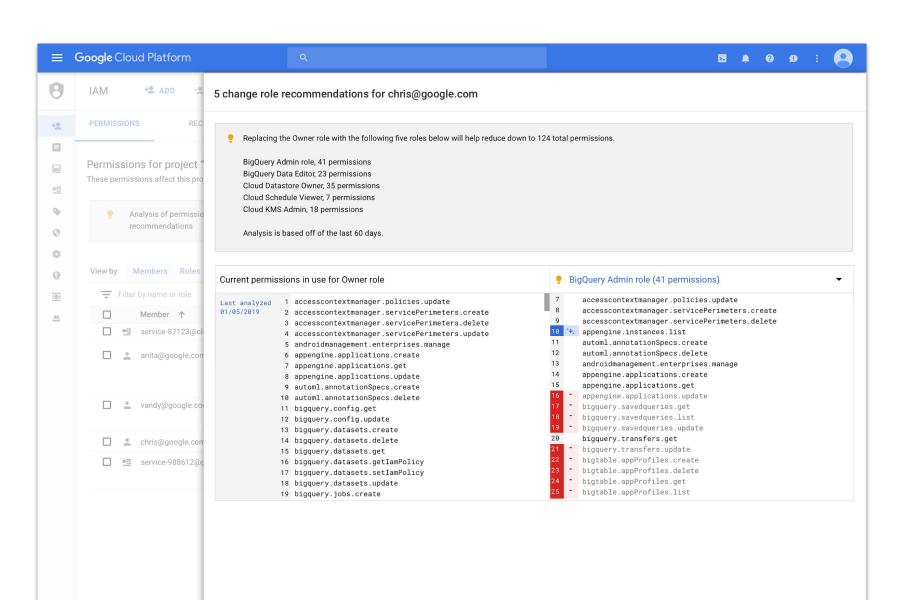
#### Stay with me. The -/+ pattern is still key here.

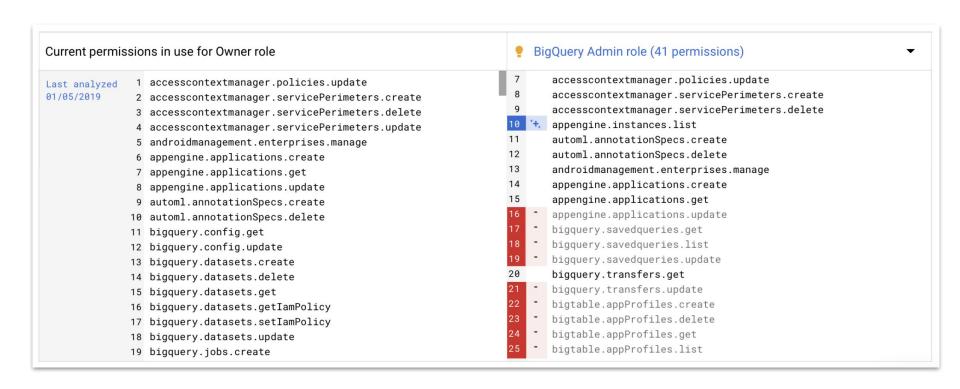


#### ...what if we added ML into

#### the traditional -/+ pattern?







10 **⁺+**₊

appengine.instances.list



#### Meet the ML sparkles

Design Patent filed GP-300272-00-US-DP on 09/16/2019

For surfacing machine learning recommendation in a code diff.

No design challenge is too small or too big to take on.

# **Key Takeaways & Reflections**

As the first team to ship a ML product within self driving cloud...



1

# Start with the user/customer first

A PM's idea is good and all, but early collaboration from the beginning can not just shape the API but the UI as well.

The whiteboarding/brainstorming session with our customers in the iteration phase garnered some of the best data.



2

# UX the API. ...as *much* as possible.

Un-engineer the API as much as possible. UX the hell out of it.

Language in the API desperately needs a UX writer, and features that are API first without any UX consultation only creates more tech debt in the future.



3

# Anything can be patented.

Yes, even icons in a pattern.



# The UI is so easy, my mom could use it!

**Customer on GCP**Name and Company redacted

(I swear they said this).



# Debuted as the keynote product at Google Cloud NEXT last year.

Many of our top customers (Snapchat, Uber, Spotify, etc) use it frequently.



### **Thank You**