A Better Auth Experience

A Technical Report submitted to the Department of Computer Science

Presented to the Faculty of the School of Engineering and Applied Science
University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science, School of Engineering

Eric He
Fall, 2021

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature _____ Date _____
Eric He

Approved _____ Date _____
Rosanne Vrugtman, Department of Computer Science

ABSTRACT

AWS Services do not currently offer a consistent, accurate and resilient Authentication and Authorization experience across all AWS services. I designed and developed a customizable dashboard to aggregate and disseminate the finding of my team to show which AWS services were not meeting IAM policy standards. I created a workflow for the AWS QuickSight dashboard using Amazon internal tools such as S3, Glue and Athena. I then automated the process with AWS CDK. The result was an internal wiki that hosted a dashboard with views for both managers and developers. The manager view was able to provide an overview of the team's progress in a snapshot, while the developer view was able to help debug and identify consistency problems. The next step to create a better Authentication and Authorization experience across all AWS services would be to support more functionality for IAM policies and automate ticketing based on service team preference.

1 INTRODUCTION

Customer Obsession is a core leadership principle at Amazon. This obsession is meant to earn and keep customer trust. One way to do this is to create a consistent user experience for the customer. Users of AWS services should expect the same results and consistency when accessing actions, resources, and condition keys for AWS IAM policies. However, this is currently not the case. Since AWS is a very large company and all service teams at AWS are treated as their own small startup, it is very hard for them to correlate such consistency among all services. It is also hard to drill down which service teams are causing a problem and identify how to help the service team to create a consistent user experience.

The first step to solving this problem was visualizing each service team's consistency. This can be used to track the progress of each AWS service and quickly notify them to fix the actions, resources, and condition context keys for their IAM policies. This process is currently done manually and automating it will save time and keep the most up-to-date data from all the AWS services.

2 RELATED WORK

Amazon provides a wide range of internal tool documentation for anything that they create. Along with the documentation provided, Amazon has its own internal version of "google" or is.amazon, "stack overflow" or amazon.sage, and "github" or code.amazon. All of these provided a great resource allowing me to read and gain knowledge about my subject. I followed many tutorials created by other Amazon engineers posted on videos.amazon and was able to join office hour sessions hosted by the individual service teams that I needed more clarification from. The tutorials and internal wikis provided all the information I needed to be successful.

3 PROJECT DESIGN

The first step in tackling such a big project was to create a design document. I was given time to create a workflow before starting the project and brainstorm potential tools that could be used. While designing the workflow I realized how many tools could be utilized to complete such an open-ended product. However, at a big company like Amazon Web Services, most tools such as Google search engine, stack overflow, and tableau have all

been replicated internally. This results in pros and cons. Pro: I have access to all internal services and the information I needed was there. Con: It takes a while to find exactly what I am looking for because the terminology is very specific. There are acronyms for everything and the whole workflow of the sites was confusing.

I wanted to use AWS QuickSight; however, it had its limitations, the main one being that it was unable to create certain views. After getting a visual of what the final dashboard could look like, I tried to replicate a simple example of it with AWS QuickSight. One restriction I found was that data from multiple databases could not be combined into the same views. This was a critical part of the dashboard. While contemplating alternatives I discovered a workaround using an AWS Lambda script to run a daily SQL queries using AWS Athena accessed through with AWS IAM permissions.

The overall workflow of the dashboard with its multitude of services is:

1. First data is grabbed from both internal API endpoints and our team's own data lake containing all the test data from each individual service team.
2. Next, the data is stored into AWS s3 buckets where they are partitioned based on date.
3. From there the data is crawled/parsed with AWS Glue and stored on a SQL database on AWS Athena.

There are multiple tables within the database for each of the different data sources. These tables were then joined using an external AWS lambda script that selected which parts of the data were needed to create a combined table with all the information that was needed to display the proper views in AWS QuickSight.

4 RESULTS

The dashboard was able to aggregate and disseminate all data lake findings of 9800 APIs spread across 270 services to assist IAM in auditing and monitoring progress. The two views that we designed were the manager view and the developer view. The manager view showed the progress of the Auth team in its attempt to create a consistent user experience. This view showed management a snapshot and trends of the team a glance. The main widgets were percentage of services covered, percentage of APIs covered, and percentage of total context keys covered.

The developer view provided multiple heatmaps that helped locate services that were not meeting standards. These services made it possible for us to dive deeper into each service and find their provided APIs and trends, along with their status on which context keys, resources, and actions they were failing. This view also allowed us to view the submitted tickets for each service team and their status.

Many service teams that were confused about why they were getting ticketed came to look at the dashboard to check their status. This helped them to quickly identify the problems with their IAM Policy support and fix them. The dashboard was embedded into an AWS internal wiki along with ample documentation for use.

5 OUTCOMES

By designing and creating a dashboard for our team, I was able to accelerate the work for other engineers on my team. I was also able to get more visibility for our team because managers could now see the process that our team was making quickly. This could lead to our team growing bigger quicker.

6 NEXT STEPS

The dashboard was created with the purpose of only showing AWS global context keys at first, however, support for both resources and actions were in the process of being implemented. Future support for any AWS policies should be shown on the dashboard to help developers and managers identify problem service teams.

The dashboard shows the trends of individual service teams, but it doesn't do anything with that information. A next step would be to have each service team set up its own notifications to constantly check if they have broken any AWS IAM policies.

Ticketing directly to AWS services teams from the dashboard has not been implemented, but this is something that can be automated to save time for developers testing each case for false positives before ticketing.

Lastly, support for the dashboard has only been implemented for North America. A cloud formation script should be tested to help replicate the dashboard for each available region so that it can be used globally.

REFERENCES

1. *Amazon athena—Serverless interactive query service—Amazon web services*. (n.d.). Amazon Web Services, Inc. Retrieved November 18, 2021, from https://aws.amazon.com/athena/
2. *AWS glue—Managed etl service—Amazon web services*. (n.d.). Amazon Web Services, Inc. Retrieved November 18, 2021, from https://aws.amazon.com/glue/
3. Quicksight vs tableau | top comparison between with infographic. (2020, December 24). *EDUCBA*. https://www.educba.com/quicksight-vs-tableau/
4. *What is amazon quicksight? - Amazon quicksight*. (n.d.). Retrieved November 18, 2021, from https://docs.aws.amazon.com/quicksight/latest/user/welcome.html