Automating Security Checks Using Cloud Tools Through Additions to an API

A Technical Report submitted to the Department of Computer Scinece

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

Jihong Min

Spring, 2022.

Technical Project Team Members

Kyle Mikolajczyk

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

Rosanne Vrugtman, Department of Computer Science

Daniel Graham, Department of Computer Science

James Cohoon, Department of Computer Science

Automating Security Checks Using Cloud Tools Through Additions to an API

Jihong Min
Computer Science
University of Virginia
Charlottesville, Virginia USA
jjm4vf@virginia.edu

ABSTRACT

Viasat, a satellite internet company, tasked its cloud engineering team, VICE, with managing and providing cloud services for the company. The main mission of VICE is to build services and platforms to provide a variety of operational and development services to teams throughout the company, primarily using Amazon Web Services (AWS). As a software engineer intern in VICE, teaming with one other intern, I made additions to the Viasat.io API in order to automate certain security checks with AWS accounts and audit AWS logs. Our changes provided the engineers with more tools for managing their resources in the cloud, making it easier for them to keep track of their AWS accounts and providing additional safeguards against fraudulent activity to improve the security of VICE's services.

KEYWORDS

Amazon Web Services (AWS), Security Automation, API

ACM Reference format:

Jihong Min. 2021. Automating Security Checks Using Cloud Tools Through Additions to an API.

1 Introduction

Checking for any vulnerabilities is crucial in maintaining the security of a system. In a team that is responsible for management of all the company's cloud accounts, efficient tools are needed to keep track of all the data. As VICE creates cloud tools for the engineers in the company, they are also figuring out the best way to make the experience reliable. As all cloud providers do, AWS has an extensive set of tools to make sure that accounts and the resources in those accounts are secure. However, there are some checks that are still insufficient and not fully personalized to VICE's use cases. Therefore, VICE elevates security through additions to the Viasat.io API. This API, a tool used by our customers, Viasat Engineers who build infrastructure and software, is one of the main parts of the team. By making additions to this API utilizing tools provided by AWS, VICE enhances the security of Viasat Cloud.

2 Background

As mentioned, the primary focus of the work was making additions to VICE's Viasat.io API. As working in the cloud can incur a lot of costs, the team wanted to create solutions that could reduce the amount of reliance on the cloud provider's tools, while also maximizing security. The goal was to expand upon existing set of cloud services, to create a cloud experience that would better fit the needs of Viasat.

3 Related Work

Salah's article, "Using Cloud Computing to Implement a Security Overlay Network" (2013), discussed ways to use these networks to provide services like intrusion detection systems, antivirus and antispam software, and distributed denial-of-service prevention [1]. Although we didn't implement a security overlay network, it was similar to our work in that it utilized cloud tools to provide additional security to different systems.

4 Project Design

VICE provided us with was a list of security checks they wanted implemented, and one API call that could be useful. The list included checking for account activity in the master and root, problematic resource configurations, and modifications of resources. The API call provided scraped every Viasat AWS accounts' AWS Config data into a Postgres database.

4.1 Using AWS Config to Implement Security Checks

Config is a tool that records all the resource configurations in an account. As a lot of the security checks on the list were checks on specific resources, this data was crucial for us to implement our additions. To be able to effectively make additions to the API, we needed to understand the code base, and how the different parts were connected. This involved reviewing already implemented API calls and specifically looking carefully at the Config data scraper provided.

Once the addition process was better understood, we needed to find out what the Config data provided us with to know what kind of function we wanted to write. This involved shelling into Postgres and trying different queries to see what information was returned and what was available. Based on what was gathered, we decided the best solution would be to write a call that would return any changes in a resource's configuration between two days.

4.1.1 Account Changes Call

The account changes call was implemented by querying all Config data separated into two arrays by day. Then, the two arrays were compared to see what changes were made between the two days. The main challenge was sorting the changes into resource deletions, additions, and modifications. And within modifications, there are instances where a specific component of a resource is removed, added, or modified. Once all the changes were sorted, it was just a matter of formatting the output in a way that the user could understand it, which we did by utilizing json objects.

To use the call, one must provide just the account ID, which would provide the changes that occurred between the two days. For additional customization, we allowed the option to provide two dates that the user wanted to compare, and also a specific resource type they wanted to check for changes to. In the output, the resource ID, resource type, and change type are provided in human readable JSON format. In the case where the change type is a modification, the type of modification, path to the changed key, and new value are provided.

By providing the option of specifying the resource type when using the call, we were able to check off a majority of the items on the list.

4.1.2 Region Check Call

Another call that was implemented using Config data was a region check. Setting up resources on AWS can be a complicated process as there are often many steps involved. Engineers are bound to make a mistake in the process at some point. For example, VICE has specific AWS regions that they primarily create their resources in. So, if a resource was mistakenly configured in an unwanted region, there was no good way to know, and the resource would continue to incur extra unwanted costs. For this purpose, we created a call that when given a list of expected regions, would return all the resources in a region that is not in that expected list.

4.1.3 Find Resource by IP Address Call

The last call we added using Config data was one that could find a resource, when given an IP address. According to many members of VICE, one of the most common questions from other Viasat engineers are about problematic IP addresses. These addresses are either causing issues, and usually they don't know where the issue is coming from or who owns it. At the time, VICE didn't have an effective way of dealing with these questions and requests.

While studying all the Config data when implementing the account changes call, we came across an IP address key value in one of the resource configurations. We didn't think much of it at the time, but we later received a request from one of the VICE engineers asking if it was possible to find IP information in the Config data. So, using the knowledge we gained, we were able to add this call so given an IP address, it would return the account ID, resource ID, and resource type that the IP is associated with. Using this tool, VICE could quickly find out whom to contact and fix any issues.

4.1.4 Automating AWS Config API Calls

These calls, especially the account changes and region check, aren't particularly useful unless they are automated. It wouldn't be practical to have an engineer manually call the API to check for issues every day. For automation, the main tool used was AWS Lambda, a function that runs in the cloud.

As working with Lambda required learning how to navigate a completely different code base from the API calls, it took some time to get it all done. The main goal of automating our calls was to check for suspicious activity in the AWS master account. This account normally shouldn't see any changes, so it was important to have some sort of alert.

We had the Lambda function run our account changes and region check calls with preset parameters on the master account. In addition, we set up a cron job so that it would run every morning, and send an email alert to VICE with details of changes in the case there are any. This gives the team the ability to quickly find out about any issues and address them as needed.

4.1.5 Adding Calls to the Viasat.io Portal

Making these calls available to all of Viasat's engineers was also a goal. Although this can be easily done by editing the permissions in the API, running them through a command line interface is not very attractive to users. To accomplish this, we added the ability for our calls to be used on the Viasat.io Portal. The portal, created by my manager, is the web interface for the API that makes various tasks easier and more approachable for users. On the portal, we created a new tab for using all of our Config API calls. With our addition, users would only have to type in the required parameters for each call to get their results. For example, for the account changes, users can just type in their account ID to check for changes in their accounts. This addition gave VICE a good place to point users to if their problems were easily fixable by using one of our API calls.

4.2 Using AWS CloudTrail to Implement Security Checks

As we got done with the Config API calls about midway through the internship, we were looking for new ways to expand upon the security checks. As Config is only able to obtain snapshots of resource configurations in an account, we needed a way to account logs with information about log in and various actions taken by a user, which were items on the checklist.

The AWS tool we found that had this ability was CloudTrail. AWS CloudTrail is a service that helps enable governance, compliance, and operational and risk auditing of your AWS account. Actions taken by a user, role, or an AWS service are recorded as events in CloudTrail. These events include actions taken in the Management Console, Command Line Interface, SDKs and APIs [2]. To obtain this data, we found that AWS has a tool called Athena, which is a database where CloudTrail logs can be stored to be queried. Once a query is submitted to Athena, the results are stored in AWS S3 buckets.

4.2.1 Setting Up an Athena Table with CloudTrail Logs

Setting up the Athena table was also a challenging process. We needed to make sure the table was created with the proper partitioning to make sure each query would not take too long and incur high costs. Also, many permissions needed to be granted to make sure each role had access to the proper resources. Luckily, AWS has extensive documentation that explains this process in detail. Completing this process, we gained an insight on how to create a database table more efficiently, and create an IAM policy to grant permissions.

4.2.2 CloudTrail API Calls

Once everything was set up, we implemented a process of three API calls to get a user the results of specific events that occurred in an account. The first call requires two dates for the range of data to look for events, and then a list of CloudTrail events. This call submits a query to Athena and outputs a query ID. Using this query ID, the user would get the status of the query through the second API call. Once they know the query finished successfully, the third call takes the same query ID, and outputs the result of the query, with information regarding the events the user asked for.

5 Results

5.1 Config API Calls

By the end of the internship, we were able to get the Config calls into production, and available to all engineers in Viasat. Although we had finished the implementation of these around halfway through the summer, it took a bit time to get it fully finalized as we continued testing. An obstacle we encountered was the differing amount of data in the dev and production environments. As production has more data, it was crucial for our complex queries to be optimized, and outputs to be within a capacity that could be handled.

The security checks added to the API proved to be very useful for the team. This was clearly shown when our Lambda function actually sent out an alert email to the team during the internship, as there were changes in the master account. Although the alert was only for small changes that IT was making, our team lead complimented us for giving a heads up on the activity so they knew the reason for the changes.

As for the effect on the rest of the engineers in Viasat, our manager noticed a spike in users accessing the Viasat.io Portal to use our added calls. This was probably due to the fact that VICE was now pointing people to the portal link whenever they got questions about problematic IP addresses.

5.2 CloudTrail API Calls

Unfortunately, we weren't able to get the CloudTrail calls into production by the end of the internship. Although we had finished the implementation, we did not have enough time for testing and a formal code review from the team leads. A major factor holding it back was the fact that VICE only had access to CloudTrail data in their own dev and prod accounts, and not any AWS accounts outside of VICE, as that would require a complicated review process with Viasat IT. Putting that aside, we left VICE with extensive documentation of the entire Athena setup process, and many comments in the code to explain our implementation. As the CloudTrail logs have a lot valuable information for security, having our calls available even for the VICE accounts would be very helpful.

6 Conclusions

In all, the API additions were beneficial to VICE and their ability to manage Viasat's cloud services, and to the engineers that use VICE's services. As developing more tools while managing all user accounts and maintaining security is a lot of work, the additions provided expansion of security checks and more tools to help in the management aspect. The Config calls provide daily checks of resource configurations in the master account and also provides all the engineers more tools that can help with managing their accounts and troubleshooting any problems. Through the deployment to production and addition to the Portal, these calls are easily accessible by anybody at Viasat to help them manage the security of their accounts.

The implementation and substantial documentation of the CloudTrail calls were completed in case VICE wanted to use them in the future. These calls provide additional security checks for VICE's accounts to ensure that unauthorized users are not accessing their accounts.

One of our goals from the beginning was to make our additions easily expandable. As VICE might later want to automate checks for more accounts, or add more events to the list to check for, implementations should make this a simple process. With that in mind, our final implementation made these things possible with just adding a line or two of code to specify the additional accounts or events.

7 Future Work

In the future, many additional tools can be utilized to further improve security and help VICE with the management of Viasat Cloud. We also implemented another API call that would allow a VICE member to edit employee emails without manually editing LDAP (or Lightweight Directory Access Protocol) as an LDAP super admin. LDAP is often used for authentication and storing information about users, groups, and applications [3]. Viasat uses this to store basic employee information. So, the API call made it so LDAP wouldn't have to be accessed to complete a simple task. This eased the work of completing email change requests for on call VICE members.

What was special about this call was that this wasn't specifically tasked for the interns, but we were able to pick up a normal ticket in VICE's workflow and turn it around quickly using the knowledge we picked up during our time there. Implementing this showed what future work could be wanted by VICE to continue their responsibilities in the future.

8 UVA Computer Science Curriculum Evaluation

The UVA CS curriculum was very beneficial in preparing me for this experience. The most helpful was the Database systems class. As most of the API calls that were added required some sort of querying, having a good foundation of databases really made the whole process of figuring out the most efficient queries very fast. Also, when creating the Athena table while implementing the CloudTrail calls, having knowledge of partitioning allowed us to avoid an obstacle of refactoring the table if the queries took too long.

REFERENCES

[1] K. Salah, J. M. Alcaraz Calero, S. Zeadally, S. Al-Mulla and M. Alzaabi, "Using Cloud Computing to Implement a Security Overlay Network," in IEEE Security & Privacy, vol. 11, no. 1, pp. 44-53, Jan.-Feb. 2013, doi: 10.1109/MSP.2012.88.

[2] Amazon Web Services, Inc. 2021. Secure Standardized Logging - AWS CloudTrail - Amazon Web Services. [online] Available at: <https://aws.amazon.com/cloudtrail/> [Accessed 20 October 2021].

[3] LDAP.com. 2021. LDAP.com. [online] Available at: <https://ldap.com/> [Accessed 20 October 2021].