Cross Domain File Transfer for AWS Internship

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 and Applied Science

Peter Shin
Fall, 2022

Panagiotis Apostolellis, Department of Computer Science

Cross Domain File Transfer at AWS Internship

CS 4991 Capstone Report, 2022
Peter Shin
Computer Science
The University of Virginia
School of Engineering and Applied Science
Charlottesville, Virginia USA
pss9fp@virginia.edu

Abstract
Current users of AWS Diode cannot source files that exist in low security domains (low-side) from a higher security domain (high-side). My project, as an intern for AWS, was to stand up a service that allow customers on the high-side to retrieve low-side files directly from the low-side repository under the condition that a mapping already is established between the high-side and low-side customers' AWS accounts. We used AWS's Cloud Development Kit in conjunction with existing cross-domain solutions to run the backend. We also used a variety of AWS services including ApiGateway, VPC, Lambda, etc. to create the service. By the end of the internship, we had built a successful but limited proof-of-concept that performs the actions of the low-side customer without the low-side customer having to take any action at all. For this project to become a viable customer-facing feature, it must get approvals from the security managers, expand on a limitation that the service has due to the existing limitations of the S3 API, as well as stand up a matching high-side service that performs most of the business logic on behalf of the high-side customer.

1. Introduction
Terrorist sightings have been caught by a consumer drone, and this drone's recorded files are saved in an arbitrary S3 bucket that has public access. A higher level government official would like a copy of these files without having to download the entire repository in which this one video file is stored. The project we developed enables a government official to be given a mapping between their S3 account on the high-side and the repository that exists on the low-side to retrieve a copy of these files and perform further analysis. This is a potential critical use case of the solution. With AWS Diode in service as it is now, its low-side users must duplicate their data repositories onto all relevant domains and sync it constantly for Diode to be usable for the average use-case of the high-side customer. Companies that require interactions with high levels of security, such as those that deal with satellites, can employ this solution to retrieve object files upon request, saving customers a lot of time and effort. The feature enabling customers to receive files on request is one that existing and potential customers of AWS Diode have requested to improve the user experience.

2. Related Works
Cross-domain solution is an important concept. In AWS's white paper on cross-domain solutions, they describe it as a means to automatically or manually control the access and/or transform of information between different levels of security domains. Their description of the multidomain data guard is important to understand as the guard inspects all data that is transmitted between different domains to ensure that the data follows a specified set of rules before being allowed to enter another domain [1]. This white paper also provides information on the general workings of Diode as a cloud-based cross-domain solution that runs on

AWS infrastructure. This information was crucial as a solution would have been impossible to build without a fundamental understanding of cross-domain services that AWS provides [2]. For the more technical aspects of the project, it was important to read and understand the S3 section of the Boto3 documentation as it allows users to interact with the AWS SDK using Python [3].

3. Project Design

The constraints significantly affected how I approached the design process. The infrastructure of this feature was a collection of many different services provided by AWS. The constraints and infrastructure together is background information in understanding the usage flow of the feature from the high-side customer's perspective.

3.1 Constraint

Several constraints that led to the final project design. The first constraint is that the repository that exists in the higher security region, to which the file in the lower security region will be copied, must already have a mapping, or a relationship, with the repository that exists in the lower security region that contains the desired file. The second constraint is that freeform text cannot be passed from the higher security region to the lower security region because this is considered a security risk. Thirdly, the system that is in place requires the payload of the HTTPS request being sent from high to low to have a specific format. For this reason, we decided it would be better to create a generic XML Schema Definition (XSD) that customers can use to create their XML payloads. This approach allowed the team to get this one generic XSD to be approved and be used by every Diode customer rather than having to have each company's XSD approved by the security risk team and installed in the system, which would have slowed the onboarding process.

3.2 Infrastructure

The project's infrastructure exists in the lower security region, and it makes up the resources and functionality of the service. The infrastructure consists of an AWS-owned internal S3 bucket, AWS Lambda functions that make up the functionality of the feature, and ApiGateway and VPC that allow this feature to be accessible but not editable by customers in the higher security region. The internal S3 bucket stores all instances of calls made to the low-side customer's S3 bucket that captures all of the filenames in that bucket along with a unique identifier to keep track of each instance. I will refer to this as a "snapshot of the bucket." The Lambda functions handle two different cases: requesting the objects in the bucket and requesting a file transfer. When the high-side customer sends a request for the objects in the low-side bucket, a response is sent back to the high-side customer that contains the filenames in the bucket as well as a unique identifier that describes this specific request (which is the same identifier that is saved in the internal S3 bucket). When the high-side customer sends a request to initiate the file transfer, the customer must specify the unique identifier and the index at which the file they desired was located in the response that they received. This request is then processed by the Lambda function to: 1) identify the correct filename from the internal S3 bucket based on the identifier and index provided in the request; and 2) initiate a low-side client on behalf of the low-side customer, and 3. initiate the transfer of the file from the low-side bucket to the high-side bucket. This Lambda function exists in a private API that is only accessible via an access method that is provided by the technical leads at AWS Diode. ApiGateway and VPC allow the high-side customer to direct their request to this private API and receive the responses from it as well.

3.3 Feature Usage Flow

The high-side customer directs an HTTPS request to the low-side service for a snapshot of the low-side bucket that the customer has a mapping to. The service then saves this snapshot that contains the filenames and the ID to the internal S3 bucket, and this snapshot is sent to the high-side customer. The high-side customer identifies the index of the file they desire and sends another HTTPS request to the service that contains the snapshot ID as well as the index. The service then identifies the filename according to the snapshot ID and the index and initiates a file transfer using the filename found with the internal AWS tool.

4. Results

What was produced at the end of my internship was a proof-of-concept that could receive requests from the low-side and initiate a file transfer to an S3 bucket that exists in the higher security region. This almost reaches the minimum viable product that demonstrates the usage of the service and displays its functionality for the high-side customer. The proof-of-concept demonstrates that this feature eliminates the effort that must be constant by the low-side customer and allows the high-side customer to not have to rely blindly on the state of the low-side bucket from which that they need a file.

5. Conclusion

The implications of this project mean safe object requests across different security regions. Government entities and contractors that employ AWS Diode can work more efficiently and quickly as the information they need will be more easily accessible, and the need for constant maintenance of repositories will be eliminated. With this proof-of-concept, users will have an easier time accessing information located in lower security regions granted that they already have permission to do so. Information is crucial and having a method to access it can help make the jobs of those who protect and discover on a national level more fluid.

6. Future Works

This project is still lacking in that it does not meet the original intentions of this feature. First, one of the crucial limitations of the current state of the project is that it is fully functional only when the S3 bucket the object is located in has up to 1,000 objects. This is due to the limitation set by the developers when creating the S3 API. Further discussion must be held on how to work around this limitation that is best for the users from a cost and efficiency standpoint as well as best for the security of sensitive information.

Second, the testing was performed completely on the lower security level such that an account on the low side was requesting information from another low-side account and transferring the requested object to the specified high-side S3 account’s S3 bucket. A set of rules (XSD) that allow the request from the high-side account to reach the service that is stood up on the low-side must be approved to take another step toward the envisioned goal of this feature before it can be released to the customers.

Finally, to make the service as user-friendly as possible, it would be a very helpful step to stand up a high-side service that works in conjunction with the low-side service. This would allow the customer to only take one step: request a file by name. Then, the high-side service would perform all of the work that originally the high-side customer would have had to do. With these steps, the proof-of-concept can become very close to a customer-facing product.

References

[1] AWS. (n.d.) What is a Cross-Domain Solution? Retrieved October 21, 2022 from https://docs.aws.amazon.com/white

papers/latest/cross-domainsolutions/ what-is-a-cross-domain-solution.html

[2] AWS. (n.d.) Connecting Cloud-to-Cloud Infrastructure Retrieved October 21, 2022 from https://docs.aws.amazon.com/whitepapers/latest/cross-domainsolutions/ connecting-cloud-to-cloud-infrastructure.html

[3] Boto3 Documentation. (n.d.) Boto3 Docs 1.26.11 Documentation Retrieved November 17, 2022 from https://boto3.amazonaws.com/v1/documentation/api/latest/index.html