Reblaze

Securely Using Amazon Web Services

Reblaze Technologies LTD

Introduction

Security on the Public Cloud

Cloud technologies offer many competitive advantages for organizations which adopt them. However, the cloud has one potential drawback: **security**.

By its nature, the cloud is globally accessible. Therefore, moving applications and private data to the cloud would seem to be very risky.

This perception is inaccurate. If the proper security measures are followed, then the cloud is no less secure than a traditional data center. Indeed, as we shall see, in many ways cloud security is actually *easier* to achieve and maintain.

The cloud platform with the largest market share is AWS (Amazon Web Services). This white paper will discuss cloud security for AWS users—both the strategies to consider and the tools that are available for implementation.

This document discusses the secure use of AWS. If you are not yet on AWS, and need some guidance on maintaining security during the migration process, see:

- Securely Migrating to AWS
- The AWS Cloud Migration Portal
- The AWS Cloud Adoption Framework

Lastly, this document is meant to be an introduction to the secure usage of AWS. Its goal is to provide an overview and conceptual understanding of achieving robust cloud security. After reading this white paper, the next step would be to dig into a more detailed and in-depth discussion of tips and tactics. For this, we recommend Amazon's "**AWS Security Best Practices**," currently accessible at:

https://d1.awsstatic.com/whitepapers/Security/AWS_Security_Best_Practices.pdf

Security and Compliance

According to the AWS Shared Responsibility Model, Amazon assumes responsibility for **Security *of* the Cloud**, i.e. securing the underlying infrastructure of AWS:

> "AWS is responsible for protecting the infrastructure that runs all of the services offered in the AWS Cloud. This infrastructure is composed of the hardware, software, networking, and facilities that run AWS Cloud services."

Meanwhile, AWS customers assume responsibility for **Security *in* the Cloud**, i.e. securing their usage of the infrastructure. This will vary, depending on the depth and breadth of that usage:

> "Customer responsibility will be determined by the AWS Cloud services that a customer selects. This determines the amount of configuration work the customer must perform as part of their security responsibilities."

In other words, AWS customers assume responsibility for securing the assets and the services they run within their accounts. The amount of work required will vary by service.

For example, EC2 requires potentially intensive maintenance (e.g., installing patches as they are issued). Other AWS products, such as Elastic Map Reduce, are semi-managed services where much of the work is done for you. And some, such as S3, require no resources to run in your account at all. Nevertheless, regardless of the maintenance that each service does or does not require, AWS users must still ensure that they configure each service properly.

A similar "shared responsibility" logic applies to compliance, which many AWS users need to maintain. Amazon notes that AWS services are compliant with many industry standards. Some of the more prominent are PCI DSS Level 1, ISO 27001, SOC 1, 2, and 3, and CSA.

This does not mean that organizations automatically inherit this compliance merely by using AWS. It only means that the underlying infrastructure is compliant, and it provides a baseline for AWS users to build upon as they seek to achieve compliance for their own products and services.

Cloud Security Versus Traditional Security

Cloud security has some similarities, but also some key differences, to security in a traditional environment (e.g., a data center).

For example, both environments include compute, network, and storage resources which must be protected. However, in traditional security, these resources exist within a perimeter, and the network's overall security is almost fully dependent on how tightly the perimeter can be locked down. Ingress through the perimeter is highly restricted; once inside the perimeter, control is much more relaxed.

The opposite is true for cloud architectures. Here, perimeters are virtual rather than physical, and they are much less important when it comes to security. Most resources are (potentially) accessible directly from the public web. Therefore, the network's security is not dependent on the lockdown of a perimeter, but instead on the correct restriction of access to each resource, at all times, in all possible circumstances.

This task is obviously more daunting than merely focusing on a perimeter. The good news is that AWS resources are designed to be used securely, and AWS provides a comprehensive set of tools to manage permissions for various kinds of resources. However, it's important to understand that this requires careful and thorough planning and implementation, throughout each account.

In the remainder of this document, we'll discuss a number of categories of security issues that should be considered:

- Account strategy
- Identity and access management
- Network resources
- Storage resources
- Compute resources and Internet-facing applications
- Automation of all the above.

AWS Account Strategy

The first security issue to consider is the AWS account itself. Every AWS user will begin with one account; the question is whether one is the optimal quantity.

Why would a user want more than one account? Multiple accounts allow for separate service usage limits and billing. They can allow for logical grouping of resources, e.g. Dev/Test/Production.

From a security perspective, multiple accounts can be advantageous because they limit the "blast radius" of potential damage from attacks. Under ideal circumstances, this would never matter, but some users might prefer having this extra airgap just in case.

The drawback to a multi-account strategy is the increased complexity that it creates. It is possible to mitigate this somewhat:

- The AWS Organizations service can be used to centrally manage your accounts.
- Multiple accounts can inherently include some of the identity and access roles that under a single account should be set up and administered separately.

However, a multi-account strategy is still the more complicated choice.

Regardless of how many AWS accounts you decide upon, there are several steps you should take for each account:

- Enable MFA (multi-factor authentication) for root login credentials. (Requiring it for **all** logins is even better.)
- Only use the root credentials for creating IAM users and roles. All other activities within an account should be done by one of those IAM users.
- Enable CloudTrail for logging and auditing. (More on this later).

IAM (Identity and Access Management)

AWS's IAM service is a crucial part of your cloud security strategy. It defines Who can do What, and When and Where they can do it.

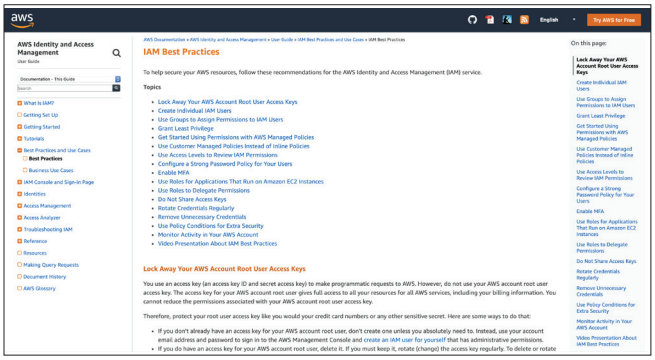
IAM allows you to define Users, Groups, and Roles.

- A User is an entity that needs to access services and resources in your account. Each entity should be a separate User. Note that Users are not limited to humans. As Amazon explains, "IAM users can be a person, service, or application that needs access to your AWS resources through the management console, CLI, or directly via APIs."
- Security permissions can be assigned directly to Users, but this is not the recommended approach. Instead, the best practice is to create fine-grained permission sets for resources, assign them to Groups, and then assign Users to the appropriate Groups. This is recommended even when it's anticipated a Group will only contain one User.
- Roles can be assumed by Users for specified times. Roles allow to grant access to various resources on an as-needed basis, without ever having to grant permanent permissions to a User or Group.

Many enterprises use authentication methods such as Microsoft Active Directory; many of these can be integrated with AWS IAM. This allows the granting of AWS IAM roles to users, or the leveraging of the AWS Security Token Service (STS) to request temporary and limited-privilege IAM credentials.

Overall, AWS IAM provides tremendous power and flexibility for controlling access to your AWS resources and services. Of the time spent creating a cloud security plan, your IAM strategy will occupy a large part of it.

Recommended Reading and Viewing



IAM Best Practices

"Become an IAM Policy Master in 60 Minutes or Less"
https://www.youtube.com/watch?v=YQsK4MtsELU

Network resources

Unlike traditional architectures, networks in the public cloud are SDNs (Software Defined Networks), so everything can be fully created, managed, and terminated via API requests. This creates both power and flexibility, and network design is an important part of a cloud security strategy.

An AWS VPC (Virtual Private Cloud) allows you to have a private and isolated networking space in the cloud. A VPC consists of one or more subnets, and is usually bound to a specific AWS region.

(AWS has multiple regions around the world, and each region has at least two Availability Zones. Each network subnet created within AWS is assigned to a specific Availability Zone. AWS customers can use this topology to create networks in multiple Availability Zones, configured to failover when outages occur, thus ensuring high availability.)

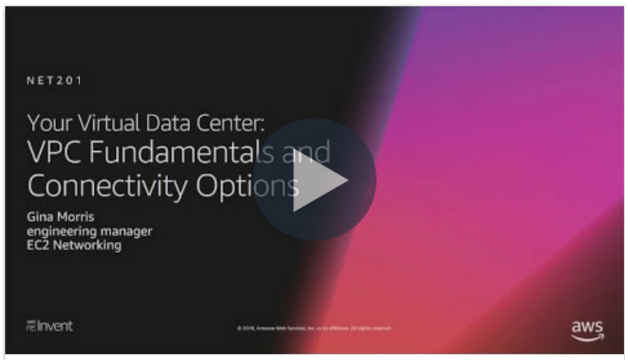
There are many ways to set up connectivity into and out of VPCs, depending on your architecture and needs. As you might expect, the best approach is to make everything private and isolated by default, then allow access only as needed. Subnets can be either private or public (with a public address space); private should be used whenever possible. Security Groups and NACLs (Network Access Control Lists) can be created to filter traffic flowing in and out of each subnet. Ideally, only specific ports are opened to traffic, rather than opening everything and then trying to control traffic with the NACL.

When Internet routing is necessary, Internet Gateways must be added. Egress-Only Internet Gateways are preferable whenever possible—they allow outbound traffic, while preventing the Internet from initiating inbound connections. Other forms of outbound privacy are provided by VPC Endpoints within many AWS services. In effect, they provide private endpoints, available only to your VPCs.

Obviously, this discussion is merely scratching the surface of what's possible using AWS network resources. A more thorough understanding can be gained by reading through the AWS deployment guide: "Building a Modular and Scalable Virtual Network Architecture with Amazon VPC: Quick Start Reference Deployment."

Recommended Viewing

"VPC Fundamentals and Connectivity Options"
https://www.youtube.com/watch?v=jZAvKgqlrjY



Storage resources

AWS provides several different ways of storing data. AWS EFS offers network file storage that's similar to a traditional NAS. AWS EBS provides block storage, similar to a traditional SAN in a data center.

A third option is AWS Simple Storage Service (S3), which offers object (binary file) storage that is not usually available in traditional environments. S3 provides unlimited, high-availability storage via a REST API, and is one of the most popular AWS services.

Unfortunately, that popularity has been a source of many problems. Because S3 is an Internet-facing service, and all objects stored within it are available via public API, it has been the source of many high-profile data leaks and other security incidents. The problem is not that S3 is inherently insecure; the problem is that many AWS users have not used it properly.

Therefore, understanding the correct use of S3 is crucial for achieving robust cloud security. Fortunately, this is not difficult.

In S3, objects are stored in buckets. Unless an object has distinct permissions assigned to it, it will inherit the permissions of the bucket in which it resides. Permissions are configured using S3 Bucket Policies, IAM Policies, or a combination of both.

By default, buckets are private, but they can be made public. Files in a public bucket inherit this "public" status (unless they have other permissions assigned), and therefore they are exposed to the public Internet. This has been the cause of many of the security incidents mentioned above.

The best and most secure approach is, unsurprisingly, to keep as many buckets private as possible. Individual objects within them that need to be accessible can have less restrictive permissions set for them, but this should be done only for specific resources and/or users. Public buckets should only be used after careful consideration, and in situations where this is truly appropriate and necessary.

As storage is used, and as new storage is created, its security can be maintained automatically. Buckets can be scanned periodically to ensure that private data is not being exposed. More on this later in the discussion on Automation.

Compute resources and your applications

AWS provides a number of different services for actually running your applications in the cloud. There is EC2 (for virtual instances), Lambda (for serverless FaaS), EKS (for managed Kubernetes), and ECS/Fargate (for serverless containers as a service).

Securing compute resources will vary depending on the type of resource. However, some security requirements are the same regardless of the underlying resource: the need to protect Internet-facing applications and APIs from attackers.

To block hostile web traffic, Amazon offers AWS WAF and AWS Shield.

AWS WAF helps customers protect their environments from SQL injection attacks and cross-site scripting attacks. It also allows customers to create custom security rulesets, consisting of various conditions which, when fulfilled, result in incoming requests being allowed or blocked.

AWS Shield provides DDoS protection for AWS customers. It has two modes: Standard (which is free), and Advanced (which is not free, and requires a service commitment).

These products perform well, and they can be one part of a good overall cloud strategy. However, it is a common mistake to rely **exclusively** on these products for web security. Each product does its job well, but they are not designed (or intended) to provide comprehensive security for the organizations using them.

The issue is that AWS WAF and AWS Shield have limited scope. They are effective at blocking attackers, but not as effective at detecting all the attackers that need to be blocked. Although they can identify simple attacks like volumetric DDoS, they cannot detect more sophisticated threats.

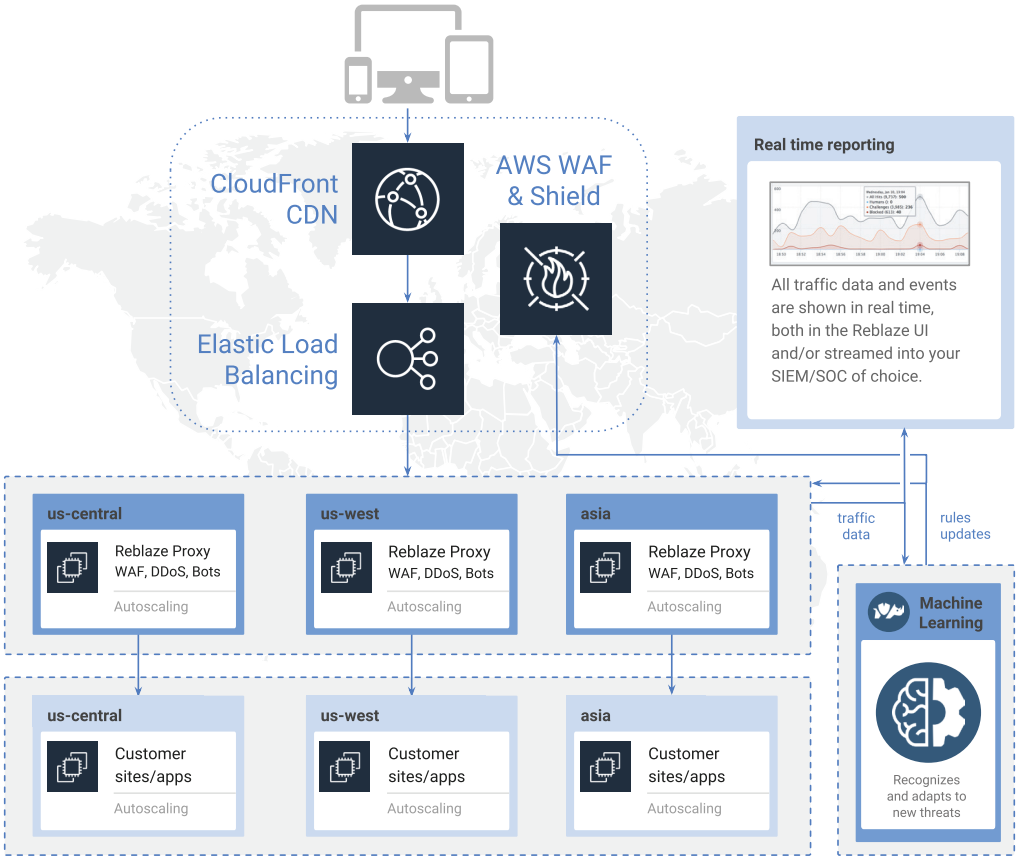
Therefore, they are meant to be supplemented by rulesets that are manually created and maintained by the users. But ruleset administration is not a trivial activity.

To use AWS WAF and AWS Shield, organizations need security-focused personnel who continually leverage log analysis tools, examine traffic request patterns, identify new sets of rules (or modifications to existing rules), test those rules, and implement them within AWS WAF. This is obviously a complicated and time-consuming process, which must be followed on a regular basis, and it lacks real-time auto updating. It potentially puts the environment at risk, as the rules are not automatically tuned to the

current traffic pattern. And doing it all correctly requires a level of web security expertise that is rare within most organizations.

The best approach for these products is to use them as an **enforcement mechanism**. For example, Reblaze is a fully managed web security solution that runs natively on AWS. In addition to a next-generation WAF/IPS and DoS/DDoS protection (both of which go beyond the capabilities of AWS WAF and Shield), Reblaze also provides advanced bot detection and management, real-time traffic control, full traffic transparency, biometric human behavioral analysis, adaptive threat detection based on Machine Learning, and many other benefits. The Reblaze platform can act as the "security engine" for AWS WAF and AWS Shield: Reblaze identifies threats, and the AWS services block the hostile traffic.

This is the recommended way to gain robust and comprehensive security from these products. It allows AWS customers to leverage the integration of the built-in AWS services, while enjoying the automated, sophisticated threat detection of an advanced platform such as Reblaze.



Architecture of Reblaze running on AWS

Automation

This document has discussed a variety of issues to consider for maintaining security while using AWS.

An important part of every good cloud security strategy is the effective use of automation. AWS includes a rich set of capabilities for automating its services, and they should be leveraged for maintaining security throughout your use of the platform.

Previously in the discussion on AWS account strategies, it was mentioned that the CloudTrail service should be enabled. This captures a log/audit trail of all API actions throughout the account. Analysis of the logs can then be automated with AWS GuardDuty, which provides intelligent threat detection and monitoring, plus several other AWS data sources such as VPC Flow Logs and DNS logs.

Another must-use service is AWS Config. It allows you to continuously monitor, audit, and evaluate the configurations of your AWS resources. It has many important uses, including security, compliance, and more. For example, you can use it to regularly scan and test S3 buckets for unwanted public data exposure.

Overall, you should use Infrastructure as Code (IaC) whenever possible throughout your AWS account. IaC is still a new trend, and many organizations have not yet adopted it. A full transition to it can indeed be somewhat disruptive. But it has numerous advantages throughout a CI/CD pipeline, not only in terms of security but also in speed and resiliency. If IaC is a new concept to you, you'll want to investigate AWS CloudFormation and all the various benefits it offers. Also see our article, "Using DevSecOps to Strengthen Security on AWS."

Conclusion

This white paper has provided a high-level overview of the issues involved when securely using Amazon Web Services.

If your organization is using, or is considering using AWS, then the next step is to construct a more detailed strategy and implementation plan. A recommended place to start is the in-depth guide mentioned previously: AWS Security Best Practices.

Securely using AWS is a front-loaded process, with much of the work done in the beginning. Much of the necessary setup and configuration is fire-and-forget. However, there is one area for which this is not true: **web security**. Blocking attack traffic that is attempting to access your applications, services, and APIs is an ongoing process, evolving constantly as the threats themselves grow and change.

Reblaze is a fully managed cloud platform, providing comprehensive, effective web security for AWS, automating its inherent security capabilities and adding many more. Machine Learning provides accurate, adaptive threat detection; dedicated Virtual Private Clouds ensure maximum privacy, performance, and protection. For more information, or to get a demo, contact us here.

Questions about the content of this white paper? Contact us at hello@reblaze.com.

To receive notifications of future publications, sign up for our newsletter by filling out the form at www.reblaze.com/contact-us/.

Reblaze Technologies, Ltd.
3031 Tisch Way
110 Plaza West
San Jose, CA 95128