Introduction to Offensive Operations in AWS

Chris Farris, PrimeHarbor Technologies, LLC
chris@primeharbor.com
April 2023

Table of Contents

Introduction

When thinking about attacking in the cloud, it's important to understand the different planes that exist.

The Network plane is what penetration testers are used to. Machines exist on a network, and Firewalls protect the machines. Machines have operating systems, and any exploit is usually in-scope for an engagement.

The Cloud Plane is new. It's the management layer that the cloud service provider provides. Access is typically via a graph or REST API, authenticated with credentials. It is a violation of a cloud service provider's terms of service to attempt to fuzz or exploit the CSP's API endpoints.

Like the Cloud Plane, the Data Plane is also authenticated with CSP-provided credentials and uses CSP-provided APIs. The primary difference is purpose - The Cloud Plane is for managing the infrastructure of a customer's cloud environment, while the data plane is for accessing data stored. Because the volume of data requests is orders of magnitude greater than management requests, CSPs log data requests differently. Because applications have broader data access requirements, there are more opportunities for insecure configurations on the data plane. The primary example is Amazon S3 - users can access S3 via HTTP or HTTPS. S3 can be authenticated or unauthenticated. Customers must manually configure S3 logging in one of two ways - S3 Access Logging is an Apache log look alike and is free. AWS CloudTrail DataEvents can also log S3 access, which costs significant money.

This whitepaper is intended to introduce these concepts to penetration testers, red teams and purple teams. It’s not intended to teach someone to be a red teamer or penetration tester, but rather to help folks in that field understand the new techniques and nuances that come from applications and infrastructure hosted in AWS.

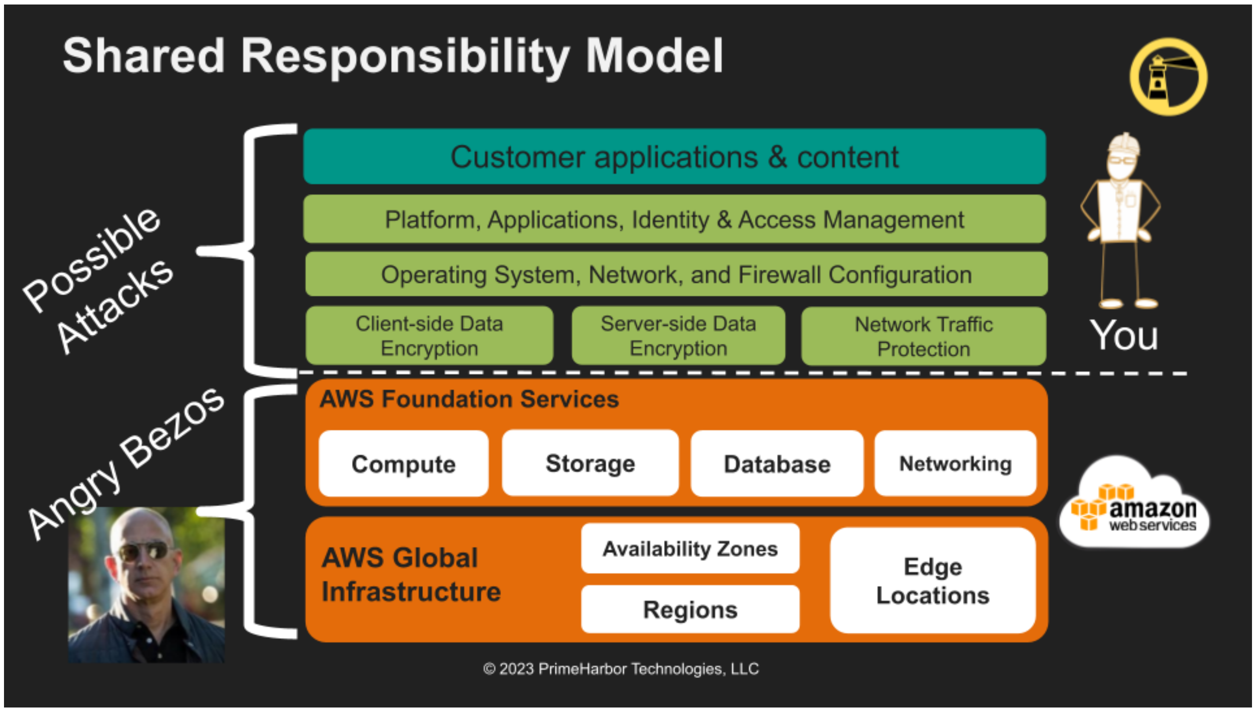
Important Notes

Shared Responsibility

It's important to note what is in scope from a terms-of-service perspective within AWS. The public cloud operates under what's known as the Shared Responsibility Model. AWS is responsible for the security of the data centers, hypervisors, and software that runs services like S3 and IAM.

As the customer, you are responsible for the operating system's security on your EC2 Instances, your application code, and how you configure the security settings on the things you create.

For a pentester or red team, the scope of your engagement should focus on things that fall on the customer's side of shared responsibility. Fuzzing the AWS API endpoints is out of scope. You can refer to the [AWS Customer Support Policy for Penetration Testing]() for a list of services and activities that are allowed, disallowed, or require AWS approval.


Shared Responsibility Model
Customer applications & content
Platform, Applications, Identity & Access Management
Operating System, Network, and Firewall Configuration
Client-side Data Encryption
Server-side Data Encryption
Network Traffic Protection
You
Possible Attacks
AWS Foundation Services
Compute
Storage
Database
Networking
AWS Global Infrastructure
Availability Zones
Regions
Edge Locations
Angry Bezos
amazon web services
© 2023 PrimeHarbor Technologies, LLC

Regions

AWS operates in 31 regions. Regions are distinct entities with their own API endpoints and (mostly) fault-tolerant from each other. Additionally, there are different AWS partitions - commercial (aws), GovCloud (gov), China (aws-cn), and an air-gapped cloud used by the US Intelligence community (aws-we'd-tell-you-but-then-we-have-to-kill-you).

One fundamental principle of AWS is that Amazon will *never* move data from one region to another unless a customer explicitly instructs AWS to do so. As an offensive operator, you need to remember to attempt these tactics against each region. Detections might be enabled for Virginia but not Singapore. VPCs might have been deleted in Ireland, but not Stockholm.

The exception to the Amazon will *never* move data from one region to another rule is with IAM. IAM data is mirrored to all of the default regions by default. If the customer enables the newer regions, IAM is also mirrored there. IAM *cannot* be mirrored across partitions. So a user's password that was created in Oregon will not be mirrored to Beijing. Note: the Hong Kong region is part of the commercial partition, not the China partition.

Initial Access

When operating in the cloud, the primary objective for an attacker is to find some form of IAM credentials. This will allow you, as the attacker, to pivot from the network plane to the cloud or data plane. In AWS, credentials come in two forms - Short Term and Long Term access keys.

Long Term Credentials are associated with an IAM User. Long-term credentials do not expire. They will remain active until the customer deactivates them. Long Term credentials consist of an Access Key ID that begins with the letters AKIA and a 40-character Secret Access Key.

Short Term credentials are associated with IAM Role. Short Term credentials have an expiration date, after which they can no longer be used to authenticate to AWS. Short Term credentials consist of an Access Key ID that begins with the letters ASIA, a 40-character Secret Access Key, and a Session Token.

The AWS CLI and most SDKs expect IAM Credentials to be in one of several places. The most common places are the shell environment and the credentials file.

The following commands set the AWS environment variables:
```
export AWS_ACCESS_KEY_ID="ASIA273IEXAMPLE"
export AWS_SECRET_ACCESS_KEY="pKZuYEOLVEMmeM..."
export AWS_SESSION_TOKEN="IQoJb3JpZ2luX2VjENT//////////...A=="
```

The following four lines are how these same credentials would be reflected in the AWS credentials file:
```
[759495629548_security-audit]
aws_access_key_id=ASIA273IEXAMPLE
aws_secret_access_key=pKZuYEOLVEMmeM...
aws_session_token=IQoJb3JpZ2luX2VjENT//////////...A==
```

There are several ways you can obtain these credentials. Long Term credentials are often found in user home directories in the ~/.aws/credentials file. They are frequently committed to code repositories. Because long-term credentials are frequently mishandled and exposed, the cloud security best practice is to avoid using long-term credentials in favor of short-term credentials.

Short Term credentials are generated by the AWS Security Token Service (STS). They're also the keys that services, like EC2, vend via the Instance Metadata service.

Ways to find Access Keys

Long-term access keys can be found in several places:

1. In source code: Developers may include AWS access keys in their source code, which can be unintentionally uploaded to public repositories such as GitHub.
2. In configuration files: AWS access keys may be stored in plain text configuration files that are readable by anyone with access to the file system.
3. In log files: AWS access keys may be inadvertently logged to system log files, which can be accessible to other users or systems.
4. In container images: AWS access keys may be included in container images that are distributed publicly, making them accessible to anyone who can access the image.
5. In environment variables: Developers may set AWS access keys as environment variables on their local machines or in server environments, which anyone with access to those systems can easily access.

If you can access a company's source code repository, you will often find access keys and other secrets in their private repositories.

You can also find access keys in public S3 buckets.

Any SSRF, Proxy-bypass, or RCE can also exfiltrate role credentials.

Evasion

The three primary AWS monitoring services are CloudTrail, GuardDuty, and VPC FlowLogs.

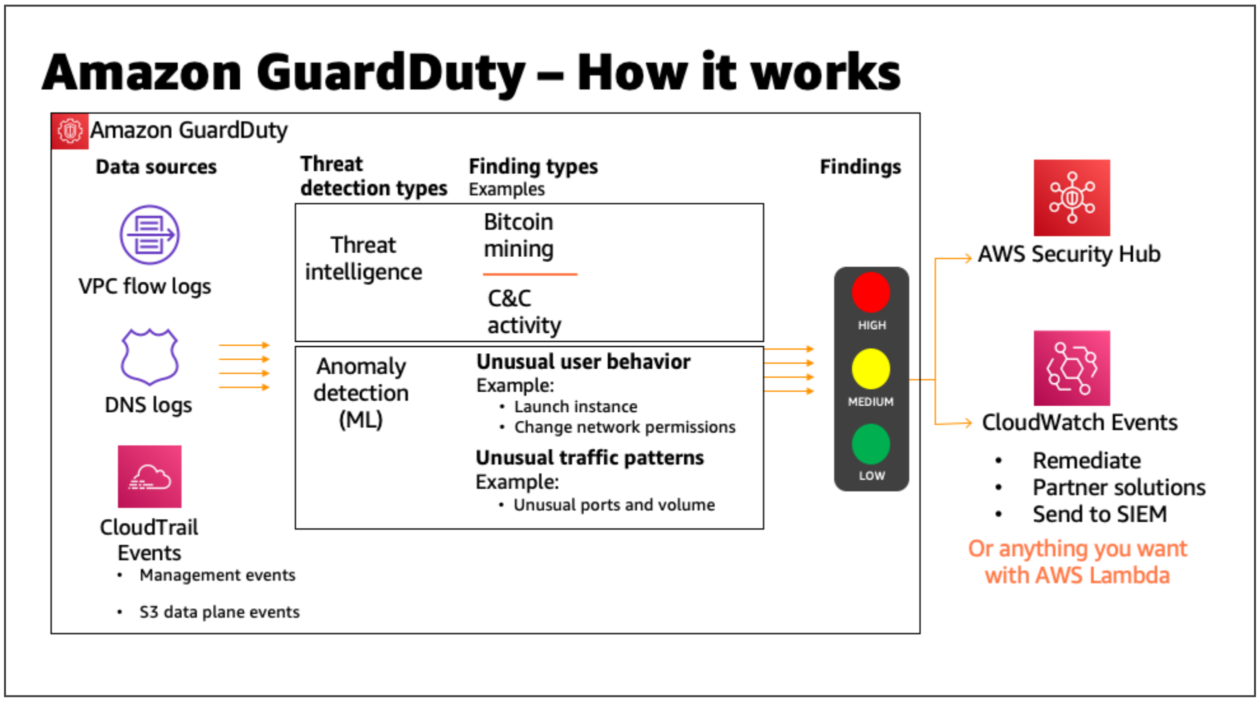
CloudTrail

CloudTrail records every properly authenticated API call made against the cloud plane and, if configured to do so, against specific services in the data plane. While in the past CloudTrail was a service that customers managed in each AWS account, more modern implementations are managed at the AWS Organization level. This makes it extremely difficult to disable CloudTrail logging. Additionally, the API calls to stop or delete logging are often monitored and usually set off alarms in organizations that monitor their API activity.

There are few if any, reliable methods to disable CloudTrail logging. Every CloudTrail event contains the Source IP Address, UserAgent, and IAM principal used to authenticate the call. The most effective evasion methods are to ensure that neither of those three elements can be attributed back to you.

GuardDuty

GuardDuty is AWS's native detection service. It leverages VPC FlowLogs, DNS Logs, and CloudTrail events to look for activity occurring from a known threat action or anomalous behavior.



Like CloudTrail, GuardDuty is often managed centrally via AWS Organizations. Compared to CloudTrail, GuardDuty is a regional service, and cloud administrators may have mistakenly not enabled it in all regions. Once enabled, it is difficult to disable, and doing so risks setting off alarms.

There are a few tactics to use for evading GuardDuty. Using IP addresses that are not in the AWS or Proofpoint threat lists is one.

When using credentials taken from the EC2 Instance MetaData Service, you will often trigger the `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.OutsideAWS` or `UnauthorizedAccess:IAMUser/InstanceCredentialExfiltration.InsideAWS` finding types. One method to evade this alert is using VPC Endpoints. Nick Frichette has written Terraform to create the endpoints in your account.

Some GuardDuty findings are based on the UserAgent in the CloudTrail event. To modify the UserAgent for AWS CLI and Python SDK calls, follow this article at Hacking the Cloud.

VPC Flow Logs

VPC Flow Logs are a NetFlow-like logging capability for all network interfaces (called Elastic Network Interfaces or ENIs) in a customer's VPC. However, it does not log network traffic for public services like S3 or IAM.

VPC Flow Logs create a record of each flow "characterized by a 5-tuple on a per network interface basis that occurs within an aggregation interval". The 5-tuple is a unique combination of source IP address, source port, destination IP address, destination port, and IP protocol (typically TCP or UDP). The aggregation interval is at a default of 10 minutes but can be set as low as one minute. VPC Flow Logs are not delivered in real-time.

VPC Flowlogs are enabled on a per-VPC basis. Logs can be written to a CloudWatch Log Group in the local account and region, or they can be written across accounts to an S3 Bucket. Because they are not centrally managed, VPC FlowLogs are the easiest to disable or delete.

Enumerating IAM Keys without detection

As documented by Nick Frichette, it is possible to obtain the username of an access key without generating a CloudTrail event. By using found access keys to attempt to publish to an SNS topic, the error from the SNS service will provide both the AWS Account ID and the Username if the access keys are still valid.

Lateral & Vertical Movement

Because of both the network and cloud plane, the public cloud introduces the concept of vertical movement. An attacker can pivot from a host to the cloud by getting session credentials, then use those credentials to vertically move back down into another host via Server Session Manager (SSM). Because IAM Roles can have cross-account trust, roles can be used to laterally move from cloud account to cloud account. Additionally, all the traditional lateral movement techniques apply to the cloud.

Ground to Cloud

Ground to cloud movement typically involves compromising a host, container, or (on rare occasions) a Lambda function to obtain IAM Credentials, which you can then use against the AWS account's control plane APIs.

The most common occurrence is obtaining credentials from the EC2 Metadata service via an RCE or SSRF by hitting the http://169.254.169.254/ endpoint inside the machine. While IMDSv2 limits the exposure of the Metadata Service from typical SSRF attacks, IMDSv2 hasn't yet been universally adopted and this technique is still used.

Other methods involve hitting an ECS Container's credential endpoint at `http://169.254.169.254/$AWS_CONTAINER_CREDENTIALS_RELATIVE_URI`, or pulling the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` environment variables from a Lambda function.

Cloud to Ground

A cloud to ground attack leverages existing AWS Credentials to compromise a host or container running in that AWS account. This would allow an attacker to pivot around the network, find a route to the on-prem networks, or persist via a listener on an exposed port.

The AWS SSM Systems Manager Session Manager (that's a mouthful) allows an AWS user to open an interactive shell on a properly configured EC2 Instance.

Another method of compromising an EC2 Instance is modifying its user data script. This script is typically run only when a machine is initially launched, but it can be configured to run on every boot. To modify user data, you must stop the machine; for the user data to be run, you must start the machine. This means that your target may notice some downtime, and if the machine is part of an auto-scaling or load-balancing configuration with health checks, it may be terminated.

Cloud to Cloud

Cloud to Cloud movement typically involves using credentials from one account to get access to another AWS account. This is typically done via Cross Account Trust policies in the destination AWS account.

CloudTrail records every instance where a user or principal in the source account assumes a role in another account. By enumerating the CloudTrail events in the source account for the AssumeRole eventName, an attacker can determine which accounts trust the source account and what targets they have for cloud-to-cloud lateral movement.

API to Console

One more thing to note about pivoting. So far, we have discussed leveraging "stolen" access keys for use with the AWS CLI. However, it *is* possible to use access keys to generate a Console session too. By calling a special federated sign-in endpoint, you can pass role credentials (access key id, secret, *and* the session token) to return a sign-in token. With that sign-in token, you can craft a URL that will log you into the AWS console as that user.

Enumeration Techniques

The cloud makes it super easy to enumerate resources. You can find all the hosts, databases, networks, and users with a few API calls. For the purposes of this missive, we shall discuss three.

Looking for Secrets

There are many places in AWS to hunt for Secrets. The first and most obvious is Secrets Manager.

```
LIST=`aws secretsmanager list-secrets --region us-east-1 \
     --query SecretList[].Name --output text`
for secret_name in $LIST; do
  echo "$secret_name: "
  aws secretsmanager get-secret-value --secret-id $secret_name \
      --query SecretString --output text --region us-east-1
done > secrets.txt
```

Another key place to find secrets is in the UserData that configured EC2 Instances. The userdata is typically an initialization shell script run on the instance when it is first created or when it boots.

```
LIST=`aws ec2 describe-instances --region us-east-1 \
    --query Reservations[].Instances[].InstanceId --output text`
for i in $LIST ; do
  aws ec2 describe-instance-attribute --instance-id $i \
     --attribute userData \
     --output text --query UserData --region us-east-1 \
     | base64 --decode > $i-USERDATA.txt
done
```

Looking for Public Stuff

The cloud is notorious for data breaches due to customers making confidential data public. While you may not need a data source to be public to exfiltrate it, why create logs and attribution if you don't have to.

AWS IAM Access Analyzer is a free tool from AWS that can show which resources are accessible publicly or to another AWS account. You can enable IAM Access Analyzer with a simple command:

```
aws accessanalyzer create-analyser --analyzer-name security \
    --type ACCOUNT
```

After a few minutes, you can review the results with this command:

```
ANALYZER_ARN=`aws accessanalyzer list-analyzers \
    --query analyzers[].arn --output text`
aws accessanalyzer list-findings --analyzer-arn $ANALYZER_ARN \
    --filter '{"status": {"eq": ["ACTIVE"]}}'
```

Note - because the resources IAM Access Analyzer analyzes are regional, you must enable IAM Access Analyzer in all regions. To get a list of all the region codes, you can use this command:

```
aws ec2 describe-regions --query 'Regions[].RegionName'
```

Looking for DataBreach stuff

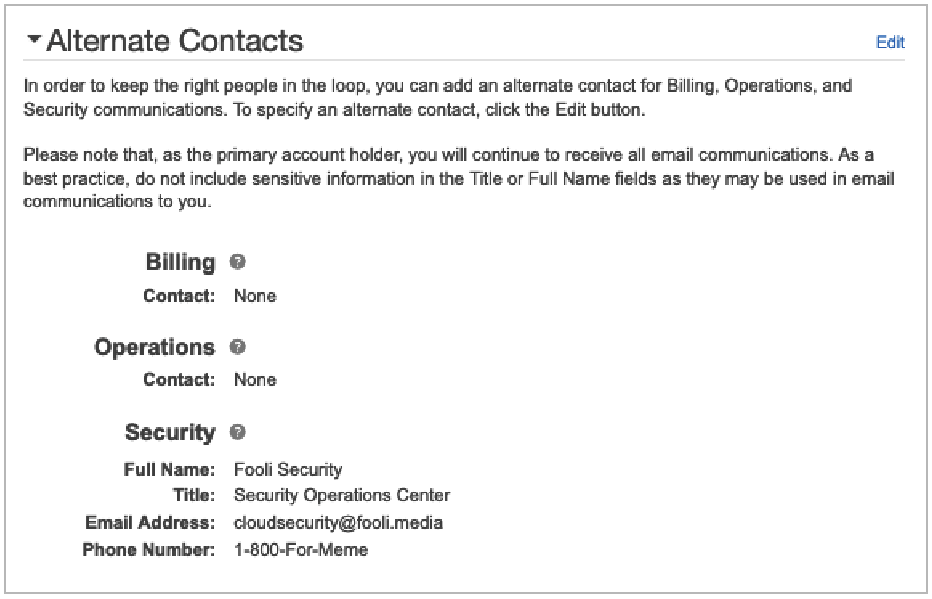
Amazon Macie is a native DSPM tool that can identify personal, financial, or credential information in S3 Buckets. You can configure it centrally via AWS Organizations or individually in each account. Macie is an expensive service, costing $1 per GB of data scanned, and as such, it probably should be out of scope for an offensive engagement. Still, if you had access to an account and needed an easy (but impactful) way to find valuable data to impact, this could be a tool in your toolkit. The cost will probably be noticed and will trigger an investigation.

Phishing

There are a few ways to leverage AWS access to find targets for phishing. The first is to look at what IAM Users may exist in the account. These are often in the form of an email address. Another method is to get the contact information for the account itself. While that doesn't reveal the root email address, it does provide the Company Name, Phone number, and mailing address.

```
$ aws account get-contact-information
{
    "ContactInformation": {
        "AddressLine1": "1205 Johnson Ferry Rd",
        "City": "Marietta",
        "CompanyName": "Primeharbor",
        "CountryCode": "US",
        "FullName": "pht-sandbox",
        "PhoneNumber": "4045383275",
        "PostalCode": "30068",
        "StateOrRegion": "GA"
    }
}
```

The final method is to see what Alternate Contacts exist on the account. AWS uses these alternate contacts to reach the billing, operations, or security contacts in case of an issue.



In order to keep the right people in the loop, you can add an alternate contact for Billing, Operations, and Security communications. To specify an alternate contact, click the Edit button.

Please note that, as the primary account holder, you will continue to receive all email communications. As a best practice, do not include sensitive information in the Title or Full Name fields as they may be used in email communications to you.

Billing

Contact:	None

Operations

Contact:	None

Security

Full Name:	Fooli Security
Title:	Security Operations Center
Email Address:	cloudsecurity@fooli.media
Phone Number:	1-800-For-Meme

```
aws account get-alternate-contact --alternate-contact-type SECURITY
```

Privilege Escalation

There are a number of methods for elevating compromised AWS permissions. Several involve using current credentials to pivot into or create new resources with more permissions. We'll outline only a few IAM specific methods:

- Overwrite a policy with a new Version (requires iam:CreatePolicyVersion)
- Set a policy to the previous version (requires iam:SetDefaultPolicyVersion)
- Creating an EC2 instance with an existing instance profile (requires iam:PassRole, ec2:RunInstances, and numerous others)
- Attaching a more privileged policy to a user/role/group you already have access too (requires iam:Attach[User|Group|Role]Policy)

- Updating the AssumeRolePolicyDocument of a more privileged role (requires iam:UpdateAssumeRolePolicy)
- Updating the code of an existing Lambda function with a more privileged role (requires lambda:UpdateFunctionCode)
- Passing a role to CloudFormation (requires iam:PassRole and cloudformation:CreateStack)
- Using iam:PassRole with a number of services, then accessing the credentials from the created resource.

Credit for this list goes to the late Spencer Gietzen of RhinoSec.

Persistence

You can use many techniques to establish persistence in an AWS account. We will only discuss a few here.

- Create an IAM User
    - Create an Access Key or Login Profile
- Create a cross-account role that trusts another compromised account
- IAM Role Juggling
    - Keep your temporary credentials active by circularly assuming multiple roles
- Insert a backdoor on an EC2 Host to expose the metadata service
- Create a lambda URL to vend credentials.

Impact

Once you have a compromised AWS account, there are three primary avenues of impact. Cryptomining is the first. An attacker will just use your credentials to spin up as many machines in as many regions as they can and run your bill way up. This is the most common form of impact for publicly exposed credentials in GitHub.

A more sophisticated attacker could engage in Ransomware or DataExfiltration (with a potential Ransomware). In the case of Ransomware, while it is possible that an attacker could "encrypt all your data", the mechanics of that are complex and time-consuming if you have a large number of S3 objects. S3 Objects are immutable, so to encrypt everything, the attacker would need to fetch the data, encrypt it, and then rewrite it back to your bucket.

A recent impact technique is leveraging Amazon's Simple Email Service (SES) to send spam. By default, AWS accounts are placed in a "sandbox" and cannot send large volumes of emails to just anyone. Some AWS accounts have production access and can send emails to anyone. Spammers will check to see if an account is out of the SES sandbox by calling the command:

```
aws sesv2 get-account --query ProductionAccessEnabled
```

If the above command returns true, you have a target that you can use for either spam or potentially phishing.

Additional Resources

- Rhino Security Labs Blog
- HackingTheCloud
- AWS CLI Command Reference
- AWS: Customer Support Policy for Penetration Testing
- AWS Security Blog: The anatomy of ransomware event targeting data residing in Amazon S3
- Firemon: What You Need to Know About Ransomware in AWS