# AWS Security

All About Securing AWS

# But first… A list of talks I attended at DefCon:

- Wiping out the Cloud
- Hacking the Cloud
- Teaching Old Shellcode New Tricks
- Assembly Language Is Too High Level
- The Black Art of Wireless Post Exploitation
- When Privacy goes Poof! And why its never coming back
- An ACE Up the Sleeve: Designing Active Directory DACL Backdoors
- MEATPISTOL, A Modular Malware Implant Framework
- Koadic C3 - Windows COM Command & Control Framework
- Game of Chromes: Owning the Web with Zombie Chrome Extensions *

  - Nearly every village except the Recon and Vehicle hacking

# Take Aways:

Blue team is eternally screwed…

I really want to learn more about Bot Nets

I was disappointed at how many people used their cell phone while there freely

A list of things I'd like to use for testing our security controls here in a controlled, planned out and approved manner. For example one of the tools… SpiderLab's Portia script.

Hiding persistence in Windows backdoors DACLs completely terrifies me.

But now to AWS….

# Shared Responsibility Model

To ensure secure services, AWS offers shared responsibility models for each of the different types of service that they offer:

- Infrastructure services: Amazon EC2, EBS, VPCs
- Container services:  RDS, and EMR
- Abstracted services: S3, SQS and SNS

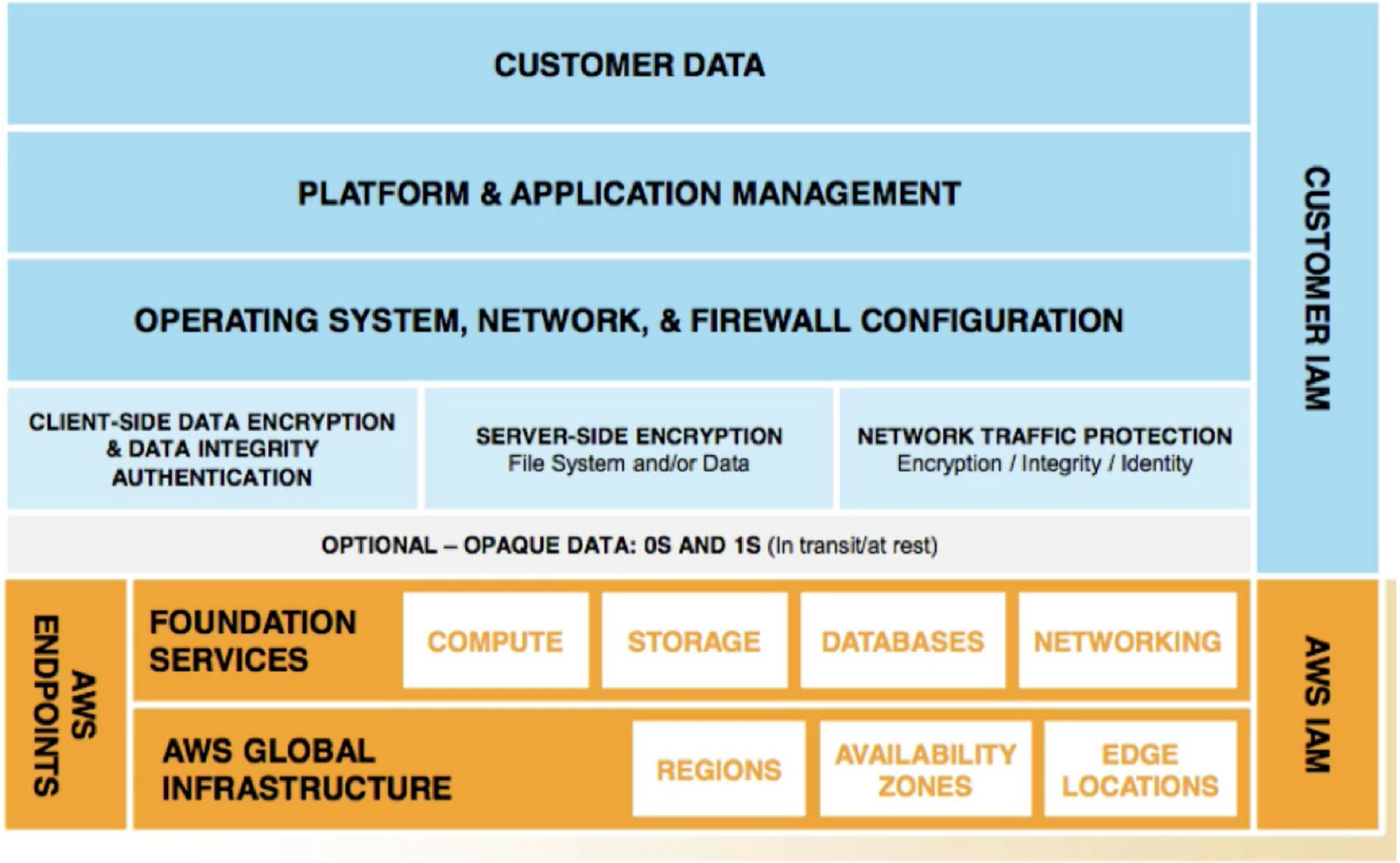
AWS provides secure infrastructure and services, while you, are responsible for secure OS, platforms, and data

# Shared Responsibility Model: Customer Responsibilities

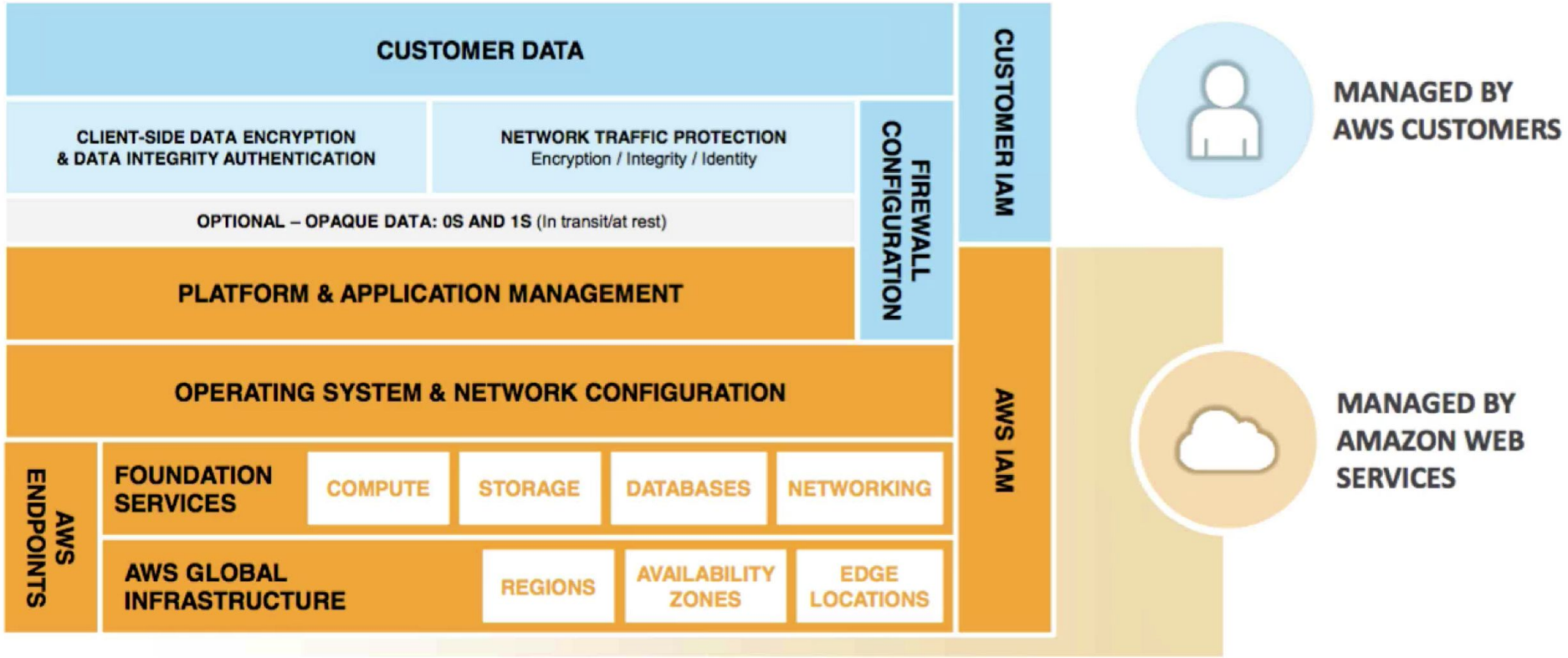
In EC2 you are responsible for the security of the following assets:

- Amazon Machine Images (AMIs)
- Operating Systems
- Applications
- Data in Transit
- Data at Rest
- Data Stores
- Credentials
- Policies and Configuration

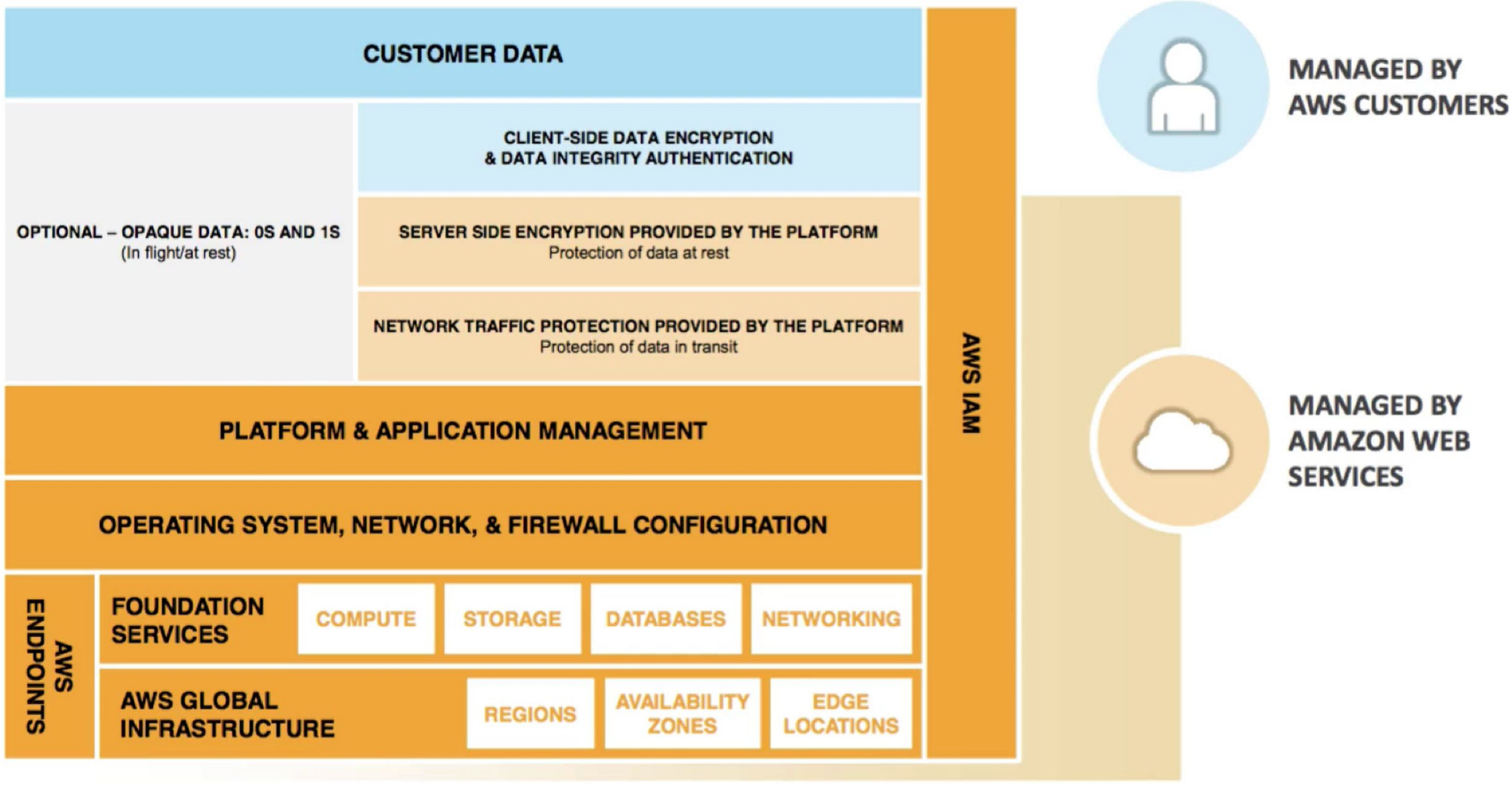
# Shared Responsibility Model for Infrastructure Services



# Shared Responsibility Model for Container Services



# Shared Responsibility Model for Abstracted Services



# IAM - Identity Access Management

With IAM, you can centrally manage users, security credentials such as passwords, access keys, and permission policies that control which AWS services and resources users can access.

By Using IAM with EC2, you can control whether users in your organization can perform a task using specific EC2 API actions and whether they can use specific AWS resources.

# Controlling Access to Amazon EC2 Resources

EC AWS IAM features allow other uses, services and applications to use your EC2 resources without sharing your security credentials

IAM Policies: Control how others use resources in your AWS account

Security Groups: Control access to your EC2 Instances

# Regions, Availability Zones, and Endpoints

AWS **regions** can be used for optimal network latency and regulatory compliance

**Availability Zones** are designed for fault isolation

Service **endpoints** are managed by AWS and provide management backplane access


OREGON
3
2
N. CALIFORNIA
3
AWS GOVCLOUD
OHIO (Coming soon)
5
N. VIRGINIA
IRELAND
3
2
FRANKFURT
UK (Coming soon)
BEIJING
SEOUL
2
2
3
TOKYO
NINGXIA (Coming soon)
INDIA (Coming soon)
2
SINGAPORE
3
SÃO PAULO
2
SYDNEY
# Region & Number of Availability Zones
New Region Coming Soon

# IAM and Controlling Access to Resources

# IAM and Amazon EC2

Organizations may have multiple AWS accounts (like ours)

EC2 enables you to specify additional AWS accounts that can use your AMIs and EBS snapshots

AMIs have a *LaunchPermission* attribute that controls which AWS accounts can use the snapshot

# Manage AWS Accounts, IAM Users, Groups, and Role

Concept of Least Privilege

**AWS Account**

- Default account that is created when you first sign up. This account is what is initially used to manage your AWS resources and services. This account has root permissions to all AWS resources and services. It is very important this account never gets used in day-to-day interactions with AWS!

# Manage AWS Accounts, IAM Users, Groups, and Role

## IAM Users

- Create multiple users using IAM, each with individual security creds and all controlled under a single AWS account
- A “IAM user” can be a person, service or application that needs access to a resource either via the management console, CLI or directly via APIs.

# Delegation Using IAM Roles and Temporary Security Credentials

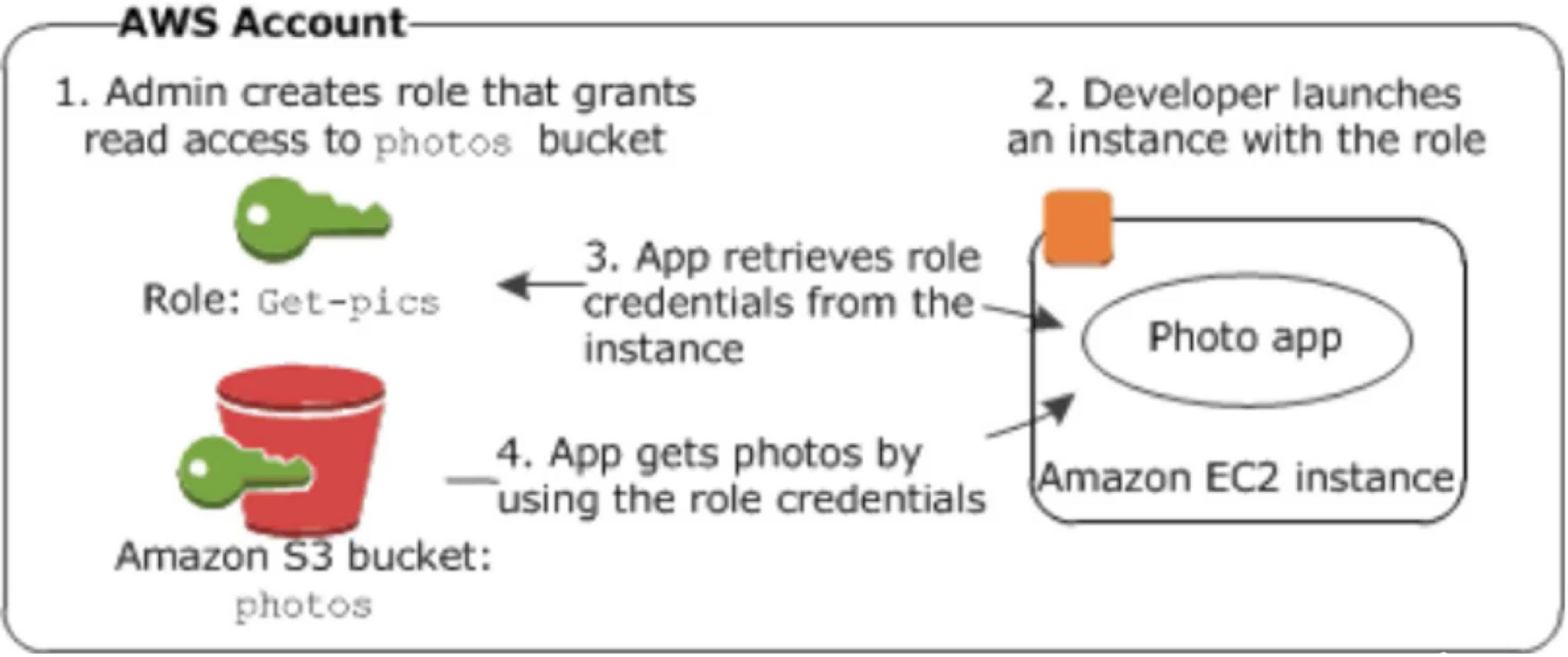
**Applications running on Amazon EC2 instances that need to access AWS resources require security credentials.**

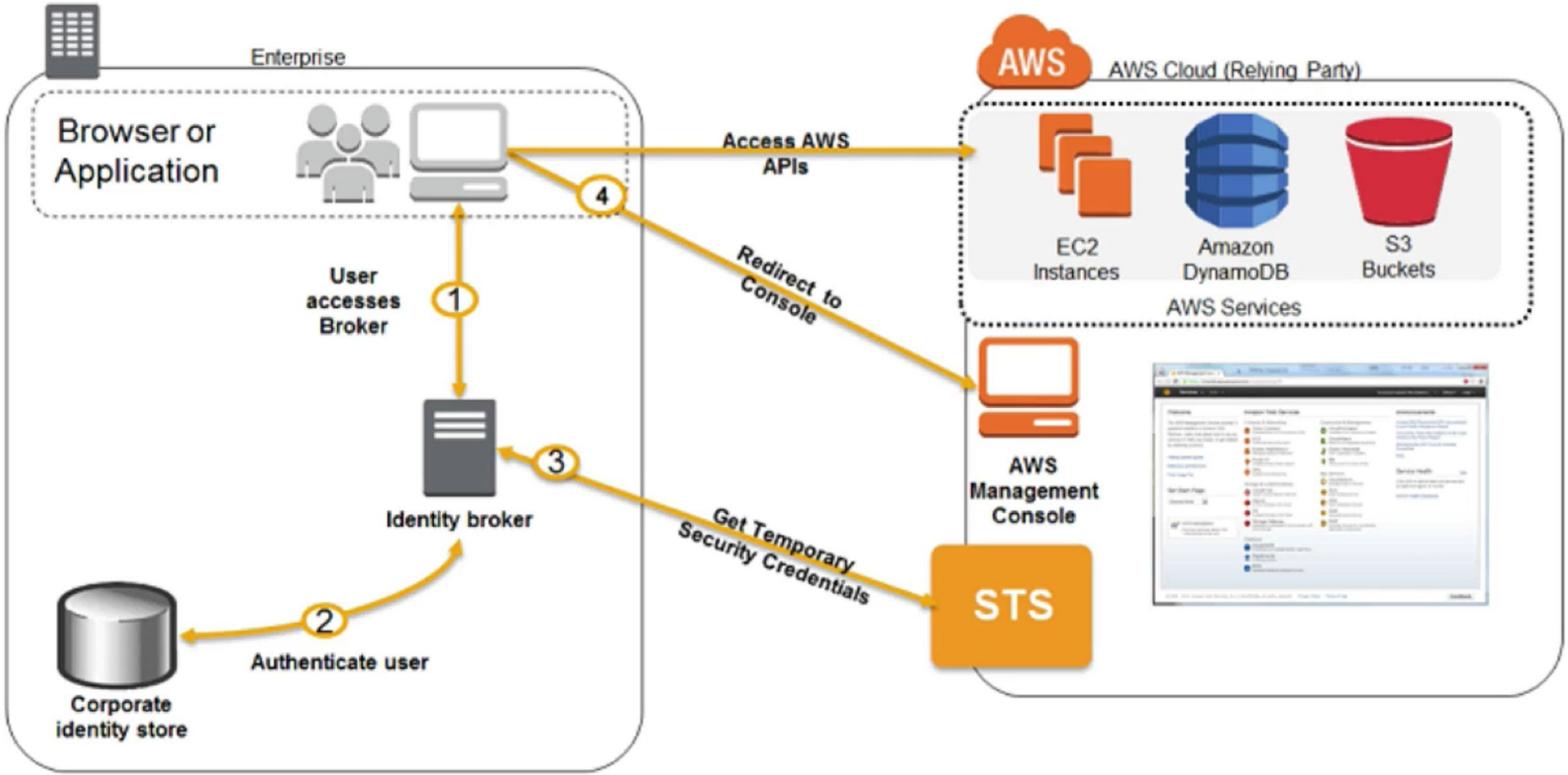
**Identity Federation**

- Provides a way to tie identities outside of AWS (ie Corporate Directory)

**IAM roles**

- define a set of permissions to access the resources that a user or service needs, but the permissions aren't attached to a specific IAM user or group.


AWS Account
1. Admin creates role that grants read access to photos bucket
2. Developer launches an instance with the role
Role: Get-pics
3. App retrieves role credentials from the instance
Photo app
4. App gets photos by using the role credentials
Amazon EC2 instance
Amazon S3 bucket: photos


Enterprise
Browser or Application
Access AWS APIs
AWS
AWS Cloud (Relying Party)
EC2 Instances
Amazon DynamoDB
S3 Buckets
AWS Services
4
Redirect to Console
User accesses Broker
1
AWS Management Console
Identity broker
3
Get Temporary Security Credentials
STS
2
Authenticate user
Corporate identity store

# Managing OS-Level Access to Amazon EC2 Instances

# Managing OS-Level Access to Amazon EC2 Instances

## EC2 Key Pairs

- Each user can have multiple EC2 key pairs, and can launch new instances using different key pairs.
- IAM uses credentials to control access to other AWS Services; EC2 key pairs control access only to your specific instance
- You can generate an import your own OpenSSH for EC2 key pair or have AWS generate it for you.

# In Linux Environments...

**Cloud-init**

- Using cloud-init when creating a new instance from a standard AMI, the public key of the Amazon EC2 key pair is appended to the initial operating system user
  ~/.ssh/authorized_keys file

# In Windows Environments...

Windows uses a service called: ec2config

On new instance creation, the ec2config service sets a new random Administrator password which gets encrypted using the corresponding EC2 key pair’s public key.

This password, along with default Admin account for the EC2 Instance can be used for authentication

# Managing OS-Level Access

## EC2 Key Management

EC2 Key pair authentication can be disabled in favor of LDAP or Active Directory Authentication.

# Protecting the OS system

**Secure Your Operating System and Applications**

- You are responsible for managing your OS and Application security

- EC2 has a web service interface which you can use to launch instances with a variety of operating systems and preloaded Applications

- Can also create your own AMIs that meet specific requirements of your organization.

# AMI Security Build Standards - Securing Linux/Unix AMIs

**Disabling Insecure Applications / Secure Services**

-Configure SSHD to allow only public key auth

**SSHD_Config:** Set **PubkeyAuthentication** to **Yes** and **PasswordAuthentication** to **No**

- Generate a unique SSH host key on instance creation

If the AMI uses **cloud-init**, this is handled automatically

**Minimize Exposure**

- Only Administrative services like SSH or RDP on start up.

**Protect Credentials and Data**

- Remove and disable passwords for all user accounts so they cannot be used to log in and do not have a default password

Run **passwd -l <username>** for each account

- Securely delete all user SSH public and private key pairs

- Securely delete all shell history and system log files containing sensitive data

# AMI Security Build Standards - Securing Windows AMIs

**Protect Credentials**

- Ensure that all enabled user accounts have new random generated passwords upon instance creation
- Ensure the Guest accounts disabled on Windows AMI
- Make sure the AMI is not part of a Windows domain

**Protect Data**

- Clear the Windows event logs
  Command line: **wevtutil cl <LogName>**

**Minimizing Exposure**

- Do not enable any file sharing, print spooler, RPC and other Windows services that are not essential but enabled by default

# Bootstrapping

- Amend and update security controls afterward

- Common bootstrapping applications include
  - Puppet
  - **Chef**
  - Capstrano
  - Cloud-inint
  - Cfn-init

Actions to consider:
- Security Software updates

# Malware: OS level infection

Understand the implications of malware infection to an individual instance as well as to the entire cloud system.

**Common Approaches for Malware Protection**

- Untrusted AMIs
- Untrusted Software / Untrusted Software Depots
    - AWS advises that you set up your own internal software depots of trusted software for your users to install and use
- Principle of Least Privilege
- Patching
- Spam
    - AWS provides special controls to limit how much email an EC2 instance can send, but you are still responsible for preventing infection in the first place
    - Avoid SMTP open relay, which can be used to spread spam, and which might also represent a breach of the AWS Acceptable Use Policy
- Host-based IDS Software
    - Popular AWS customer host based IDS software is the open source product OSSEC

# The Important Stuff:  Securing Our Data

# Securing Data on AWS

**Resource Access Authorization**

Few permission types used by AWS:

- Resource Based
- User (Capability) Based
- IAM Policies

Note: These policies can be cumulative!

# Resource Access Authorization

**Resource Policies**

- There may be cases where a user creates resources and wants other uses to access those resources.  In this model, policy attached is attached directly to the resource.
- Can grant users explicit access to manage permissions on a resource

One disadvantage is that not all services support resource-based policies.  AWS services that support resource-based policies are:

- S3 Buckets
- Simple Notification Service (SNS)
- Simple Queue Service (SQS)

# Securing Data on AWS

**Resource Access Authorization**

**Capability Policies**

- Capability Policies or user-based permissions are assigned to an IAM user either directly or indirectly using an IAM group
- Can also be assigned at run time as a “role”

**IAM Policies**

- Can be used to restrict access to a specific source IP address range or by date/time

# Policy Config example:

```
{
    "Version": "2008-10-17",
    "Id": "Policy1346097257207",
    "Statement": [
        {
            "Sid": "Allow anonymous upload to /incoming",
            "Effect": "Allow",
            "Principal": {
                "AWS": "*"
            },
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::[your_bucket]/incoming/*"
        }
    ]
}
```

# AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see key concepts in Using AWS Identity and Access Management. Here are sample policies.

## Step 1: Select Policy Type

A Policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an SNS Topic Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

**Select Type of Policy** S3 Bucket Policy

## Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a description of elements that you can use in statements.

**Effect** ◉ Allow ○ Deny

**Principal**

Use a comma to separate multiple values.

**AWS Service** Amazon S3 ☐ All Services ('*')

Use multiple statements to add permissions for more than one service.

**Actions** -- Select Actions -- ☐ All Actions ('*')

**Amazon Resource Name (ARN)**

ARN should follow the following format: arn:aws:s3:::<bucket_name>/<key_name>.
Use a comma to separate multiple values.

Add Conditions (Optional)

Add Statement

## Step 3: Generate Policy

A *policy* is a document (written in the Access Policy Language) that acts as a container for one or more statements.

**Add one or more statements above to generate a policy.**

# Protecting Data at Rest

**Accidental information disclosure**

- Designate data as confidential and limit the number of users who can access it through use of permissions for services such as S3.
- Use encryption on confidential data on EBS or RDS

**Data Integrity Compromise**

- To ensure that data integrity is not compromised through deliberate or accidental modification, use resource permissions to limit the scope of users who can modify the data
- Perform data integrity checks, such as Message Authentication Codes SHA-1/SHA-2, digital signatures, or authenticated encryption (AES-GCM)

# Protecting Data at Rest

**Accidental Deletion**

- Always verify the correct permissions and rule of least privilege is implemented
- Enable MFA Delete on services such as S3.
- Ensure Versioning is functioning properly in the case of S3 buckets.

**System, Infrastructure, hardware, or software availability**

- Restoring from Data backups in case of a system failure or natural disaster
- Services such as S3 provide data replication between Availability zones within a region

## Versioning

- (•) Enable versioning
- ( ) Suspend versioning

This suspends the creation of object versions for all operations but preserves any existing object versions.

Cancel Save

## Logging

Set up access log records that provide details about access requests.

Learn more

Disabled

## Static website hosting

Host a static website, which does not require server-side technologies.

Learn more

Disabled

## Advanced settings

### Tags

Use tags to track your cost against projects or other criteria.

### Transfer acceleration

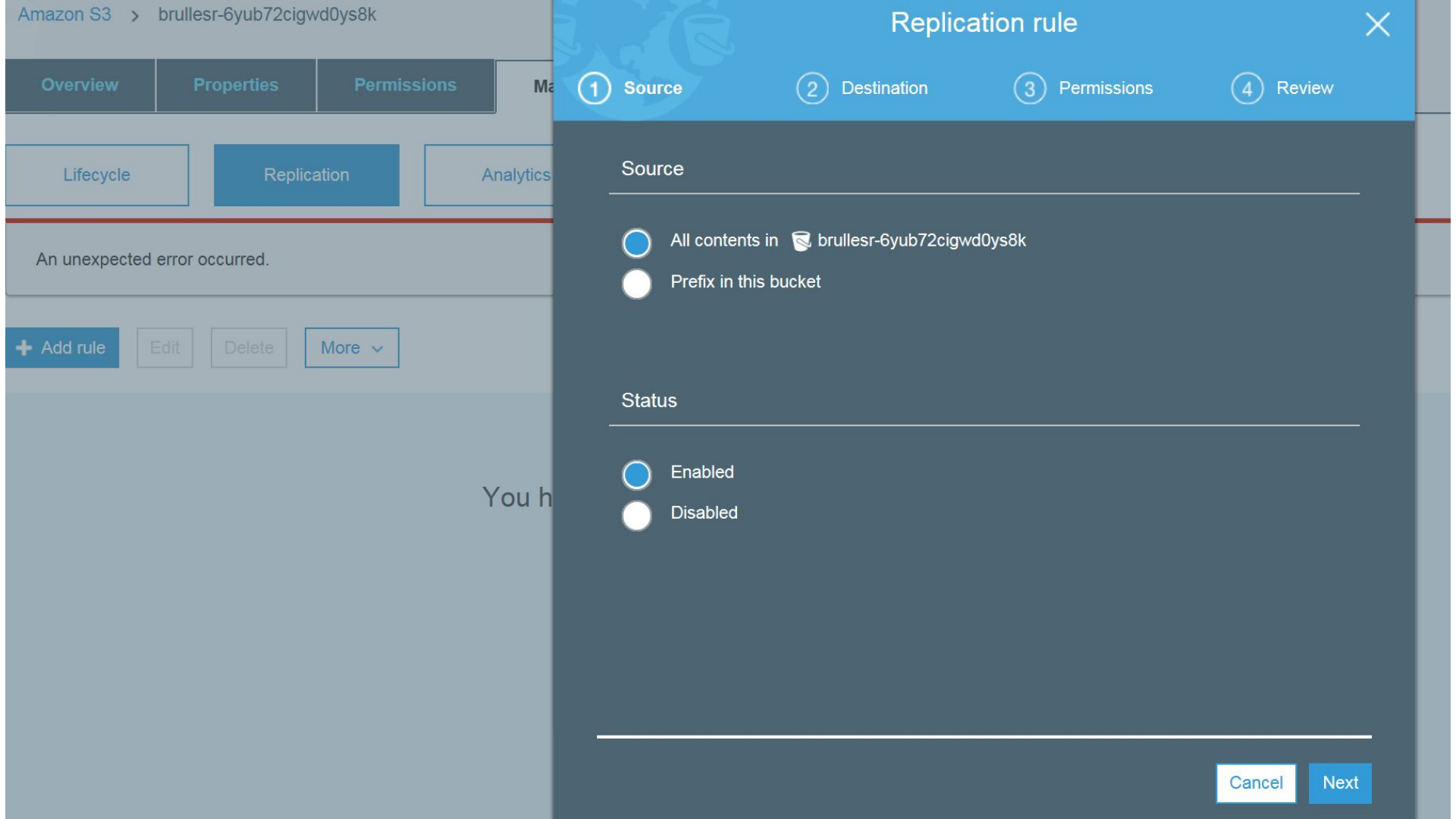
Enable fast, easy and secure transfers of files to and from your bucket.

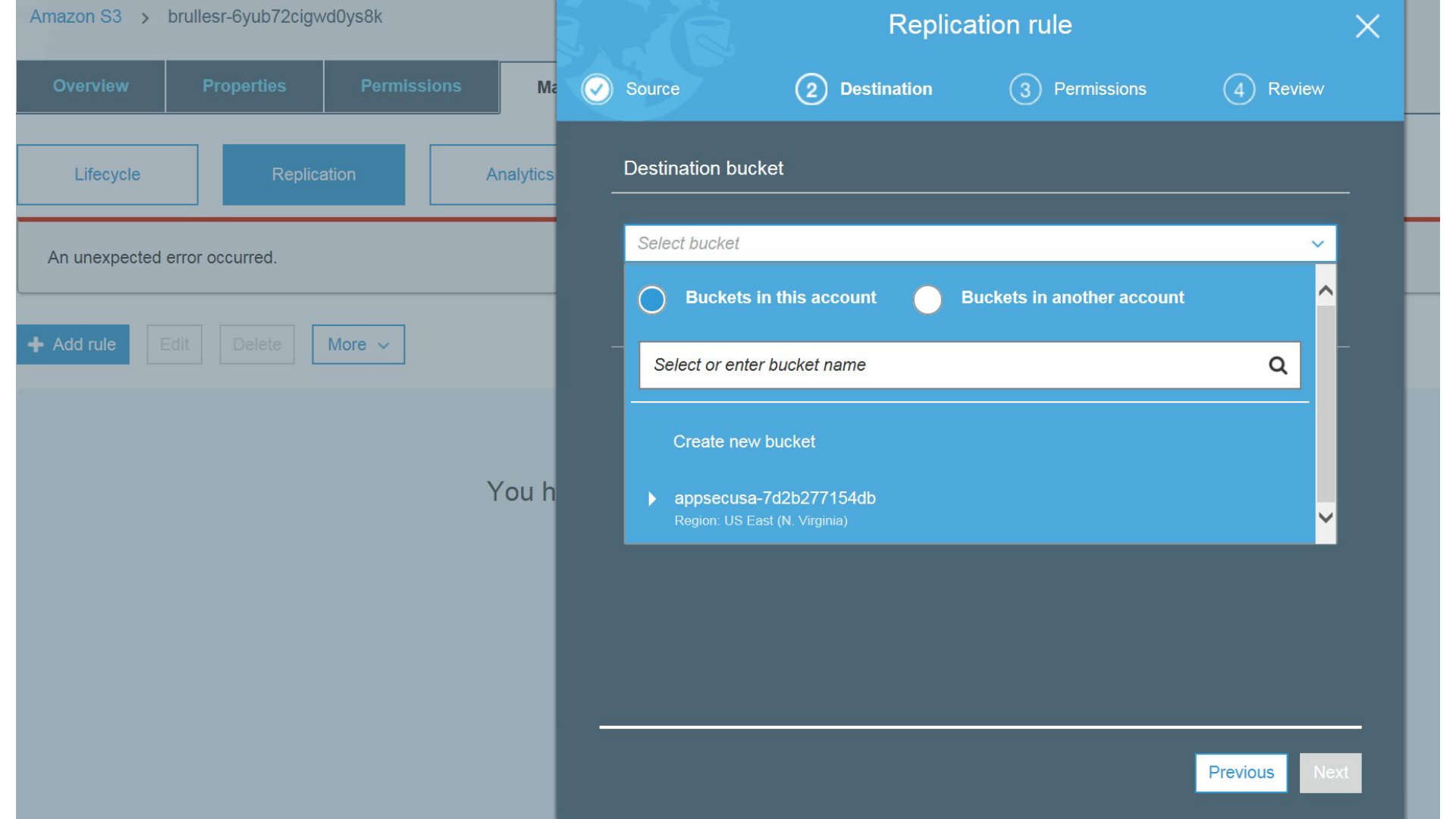
### Events

Receive notifications when specific events occur in your bucket.

### Requester pays

The requester (instead of the bucket owner) will pay for requests and data


Amazon S3 > brullesr-6yub72cigwd0ys8k
Overview
Properties
Permissions
Lifecycle
Replication
Analytics
An unexpected error occurred.
Add rule
Edit
Delete
More
You h
Replication rule
1 Source
2 Destination
3 Permissions
4 Review
Source
All contents in brullesr-6yub72cigwd0ys8k
Prefix in this bucket
Status
Enabled
Disabled
Cancel
Next


Amazon S3 > brullesr-6yub72cigwd0ys8k
Overview
Properties
Permissions
Lifecycle
Replication
Analytics
An unexpected error occurred.
Add rule
Edit
Delete
More
You h
Replication rule
Source
2 Destination
3 Permissions
4 Review
Destination bucket
Select bucket
Buckets in this account
Buckets in another account
Select or enter bucket name
Create new bucket
appsecusa-7d2b277154db
Region: US East (N. Virginia)
Previous
Next

# Protecting Data at Rest

S3 - Data Storage that is scalable, reliable and redundant

- S3 is used for storing (AMIs) which can be used for launching EC2 instances.
- S3 can store snapshots for quick data recovery

EBS - Elastic Block Store

- Highly available, can be attached to any running instance within same Availability zone.

# Protecting Data at Rest - S3

- Replication
  - S3 replicates objects across all availability zones
  - Replication is great for data and service availability, but provides no protection against accidental deletion or data integrity compromise.
- Backup
  - S3 data replication and versioning instead of automatic backups
  - Use application-level technologies
- Encryption-Server side

# Protecting Data in Transit

**Common Concerns to Communication Over Public Links**

**Accidental Information Disclosure**

- Access to confidential data should be limited
- To protect this communication, encrypt data in transit using IPSec and/or SSL/TLS

**Data Integrity Compromise**

- Authenticate data integrity using IPSec ESP/AH and/or SSL/TLS
-

**Peer Identity Compromise/ Identity Spoofing / Man-In-The-Middle**

- Recommended: use IPSec with IKE with pre-shared keys or X.509 certificates to authenticate the remote end
- Alternatively, usie SSL/TLS with server certificate authentication based on the server's common name (CN) or Alternative Name (AN/SAN)

# Real World Examples of AWS Compromise

https://threatpost.com/four-million-time-warner-cable-records-left-on-misconfigured-aws-s3/127807/

One text file, “User Profile Dump, 07-07-2017,” contained more than four million TWC customer records, most which stem from the company’s MyTWC app, an app that lets customers pay bills, upgrade services, and access voicemail, channel listings, and WiFi settings.

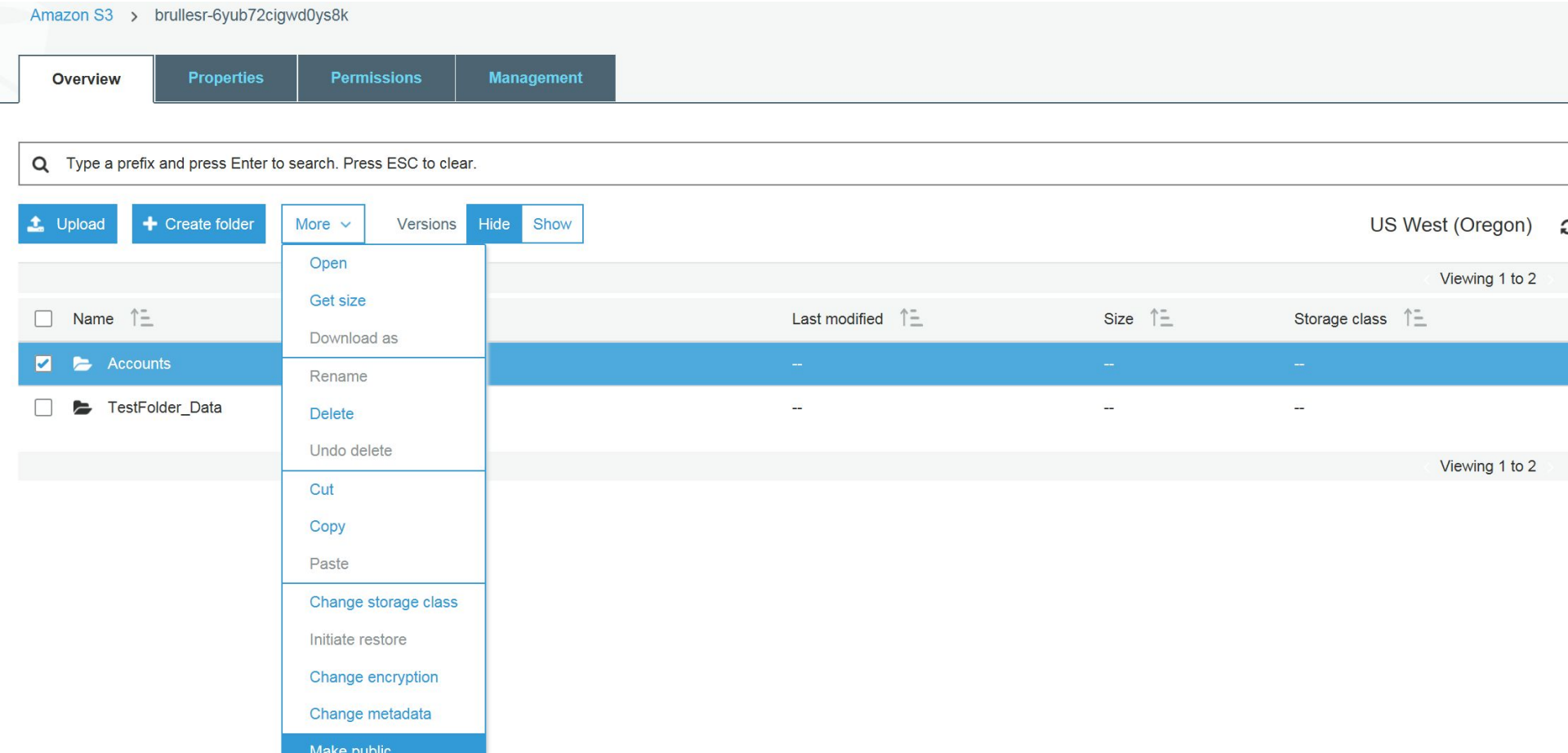
The information was left on two misconfigured Amazon Web Services S3 buckets. The buckets, while usually protected by default, had been configured to allow public access, perhaps researchers suggest, for use as a backup testing ground.

https://threatpost.com/military-contractors-vendor-leaks-resumes-in-misconfigured-aws-s3/127803/

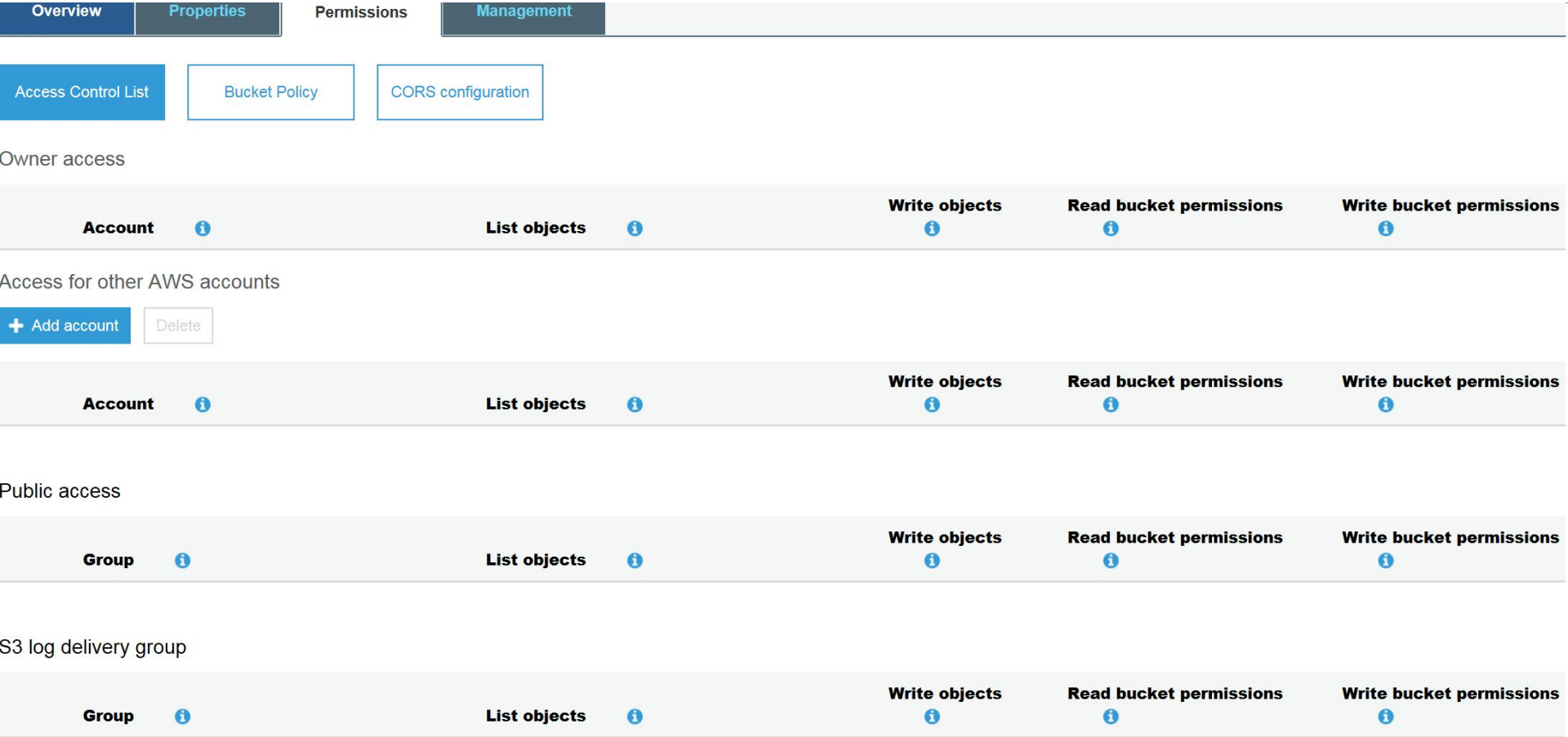
Thousands of resumes and job applications containing the personal information of U.S. veterans, many with top secret clearances, and law enforcement officers were left exposed in an Amazon Web Services S3 bucket.

That changed last week when UpGuard went public with its findings, and TigerSwan soon realized that TalentPen had used an S3 bucket configured for public accessibility to transfer the resumes and personal data to TigerSwan and had never deleted the S3 instance. AWS S3 buckets are configured private by default, meaning that TalentPen changed that setting to public for some reason.

# S3 Buckets: an inside look



# S3: Permissions



# “Murder in the Amazon Cloud”

Code Spaces was a company that offered developers source code repositories and project management services using Git or Subversion.

“It became clear that so far no machine access had been achieved due to the intruder not having our private keys.” Code Spaces said it changed its EC2 passwords, but quickly discovered the attacker had created backup logins, and once recovery attempts were noticed, the attacker began deleting artifacts from the panel.

**Separation of services! - MFA Authentication!**
Code Spaces had replicated services and backups, but those were all apparently controllable from the same panel and, thus, were summarily destroyed.

# THE END

# Overtime!!
# Detecting and Preventing Unauthorized Access

# Intrusion Detection and Prevention

Detecting Unauthorized Access

Types of Access

- Credentials
- Publicly accessible resources
- Cross account access

# Intrusion Detection and Prevention

**Detecting Unauthorized Access using Credentials:**

Types of Credentials
- Login profile
- Access key
- X509
- CloudFront
- Temporary Security Credentials
-

Where these types of credentials get used from:
- Root account
- IAM users

# Intrusion Detection and Prevention

Potentially publically accessible resources

- S3 Buckets
- S3 Anonymous Objects
- SQS Open / Public Queues

On Cross account resources, that can be made public:

- S3 Buckets
- SQS queues
- SNS topics

You want to look out for policies that could be used to gain all access (IAM APIs)
AWS Policy Generator

# Write Once Storage

# Intrustion Detection and Prevention

**Write-Once Storage with S3**

Good for things such as:

- Configuration audits
- Logs
- Tripwire data

Integrity for records of activity, historical configurations.  This can be further enhanced by moving off system or limiting availability to a select few.

We can configure Write-Once Storage by:

- Use Bucket verisoning
- MFA delete


Bucket Policy Editor
Cancel
Policy for Bucket : "writeonce"
Add a new policy or edit an existing bucket policy in the text area below.
{
"Version": "2008-10-17",
"Id": "Policy1382581126724",
"Statement": [
{
"Sid": "Stmt1382581121416",
"Effect": "Allow",
"Principal": {
"AWS": "arn:aws:iam::923022406781:root"
},
"Action": [
"s3:GetObject",
"s3:PutObject"
],
"Resource": "arn:aws:s3:::writeonce/*"
}
]
}
AWS Policy Generator | Sample Bucket Policies
Save
Delete
Close