CloudTrail Logging Internals

A methodology for investigating AWS security incidents

Eliav Levy, Lead Security Researcher @ Hunters

Hunters

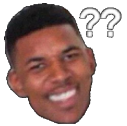
aws sts get-caller-identity

- Eliav Levy
- Security Researcher (@Hunters)
- Hobbies
  - Playing piano
  - Rockhounding

youtube.com/eliavlevy

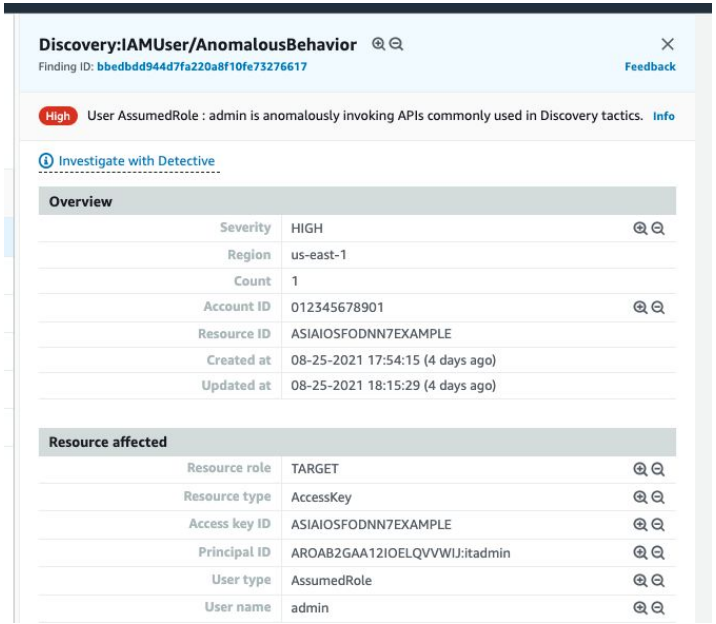
Agenda

- Investigating an incident
  - What questions to ask
  - How to ask those questions
  - CloudTrail internals
- Fun examples of CloudTrail-complexity



Hunters

Let’s start with an incident



Discovery:IAMUser/AnomalousBehavior

Finding ID: bbedbdd944d7fa220a8f10fe73276617 Feedback

High User AssumedRole : admin is anomalously invoking APIs commonly used in Discovery tactics. Info

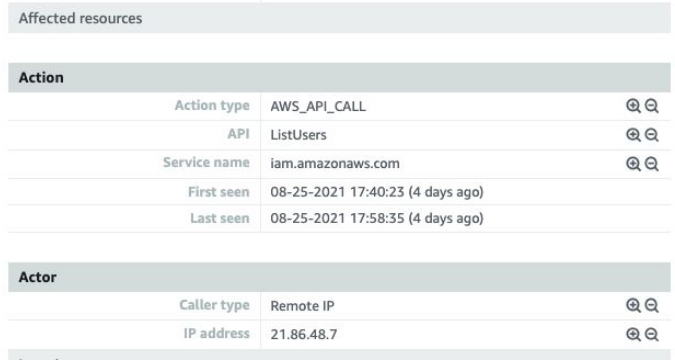
Investigate with Detective

Overview

Severity	HIGH
Region	us-east-1
Count	1
Account ID	012345678901
Resource ID	ASIAIOSFODNN7EXAMPLE
Created at	08-25-2021 17:54:15 (4 days ago)
Updated at	08-25-2021 18:15:29 (4 days ago)

Resource affected

Resource role	TARGET
Resource type	AccessKey
Access key ID	ASIAIOSFODNN7EXAMPLE
Principal ID	AROAB2GAA12IOELQVVWIJ:itadmin
User type	AssumedRole
User name	admin



Affected resources

Action

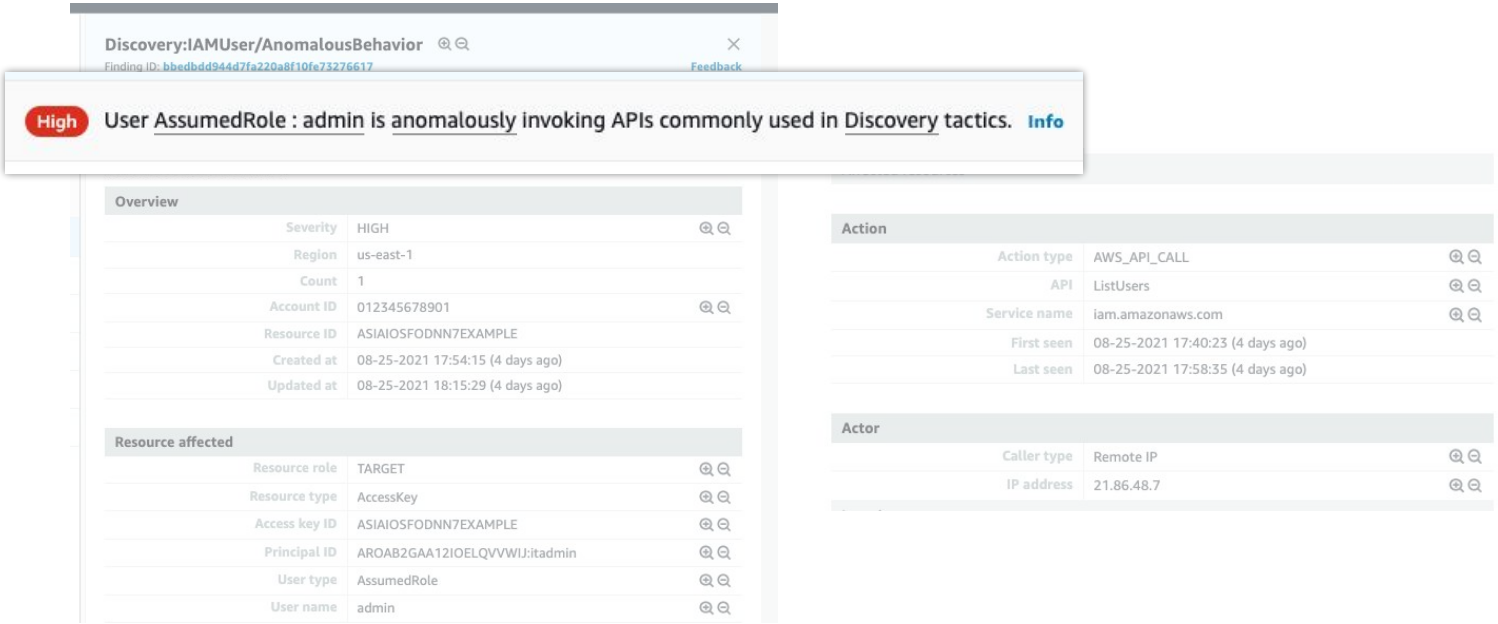
Action type	AWS_API_CALL
API	ListUsers
Service name	iam.amazonaws.com
First seen	08-25-2021 17:40:23 (4 days ago)
Last seen	08-25-2021 17:58:35 (4 days ago)

Actor

Caller type	Remote IP
IP address	21.86.48.7

Hunters

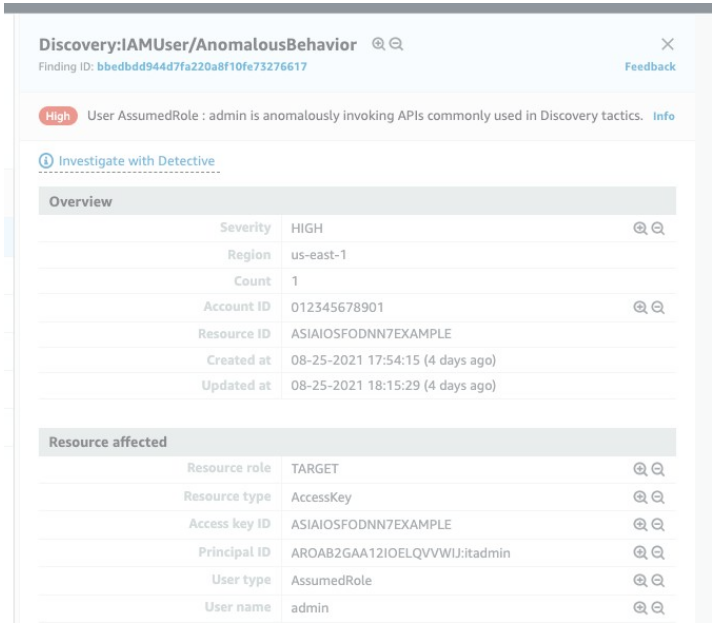
Let’s start with an incident

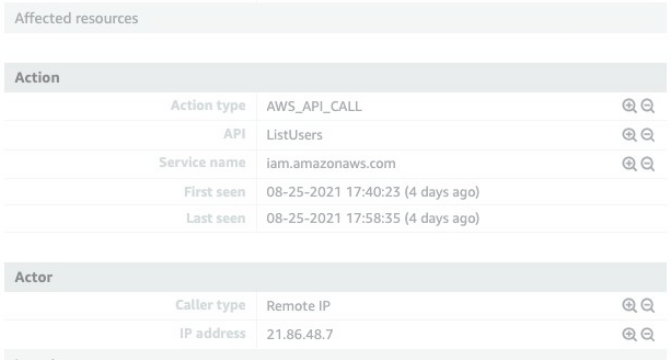


Hunters

Basic triage questions

1. Who did it?
2. What did they do?
3. From where?

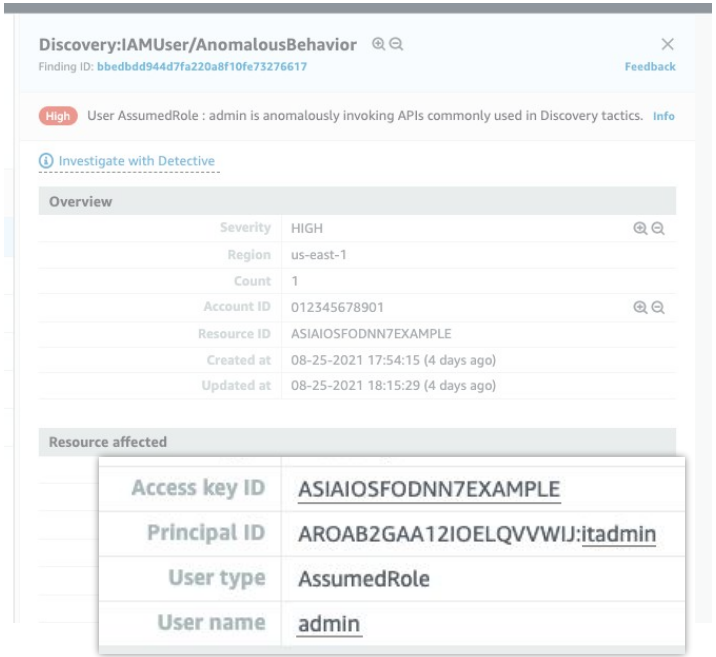


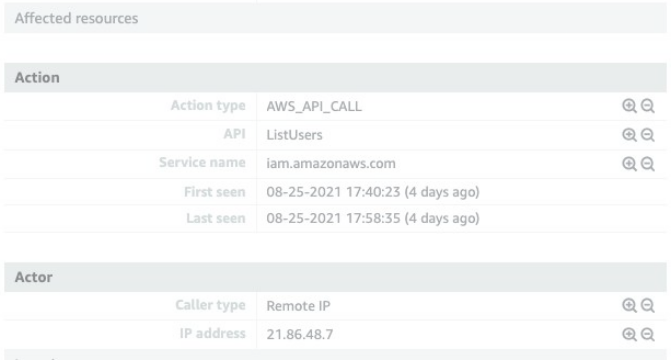


Hunters

Basic triage questions

1. **Who did it?**
2. What did they do?
3. From where?

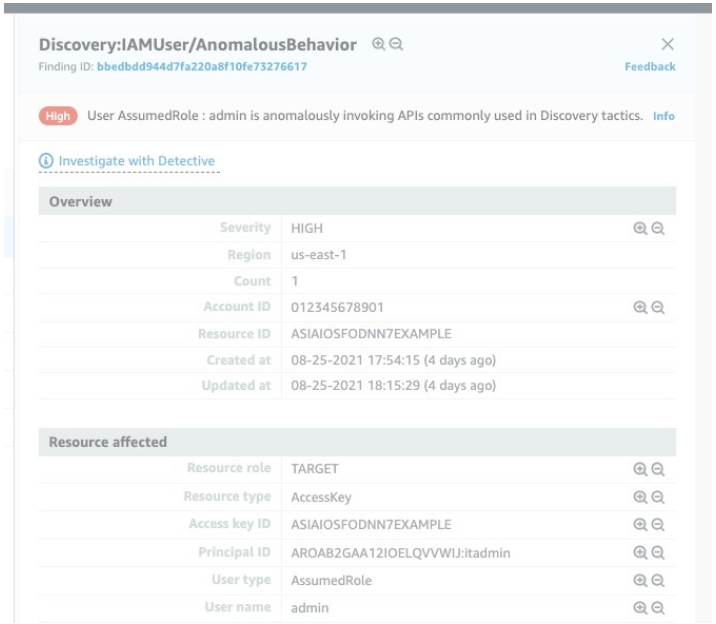


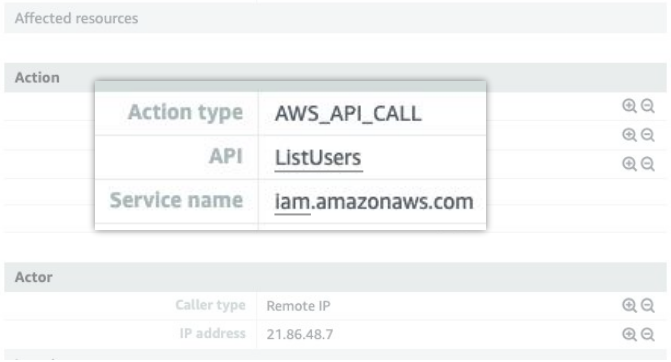


Hunters

Basic triage questions

1. Who did it?
2. **What did they do?**
3. From where?

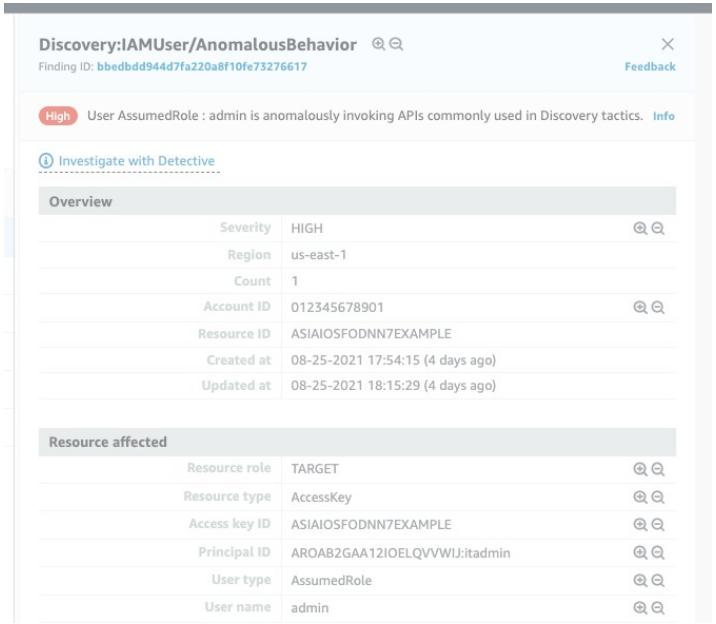


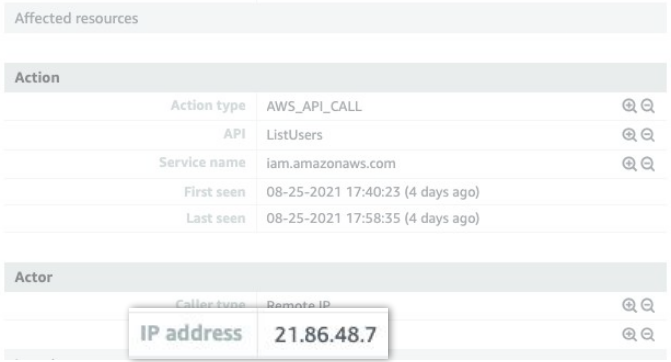


Hunters

Basic triage questions

1. Who did it?
2. What did they do?
3. **From where?**



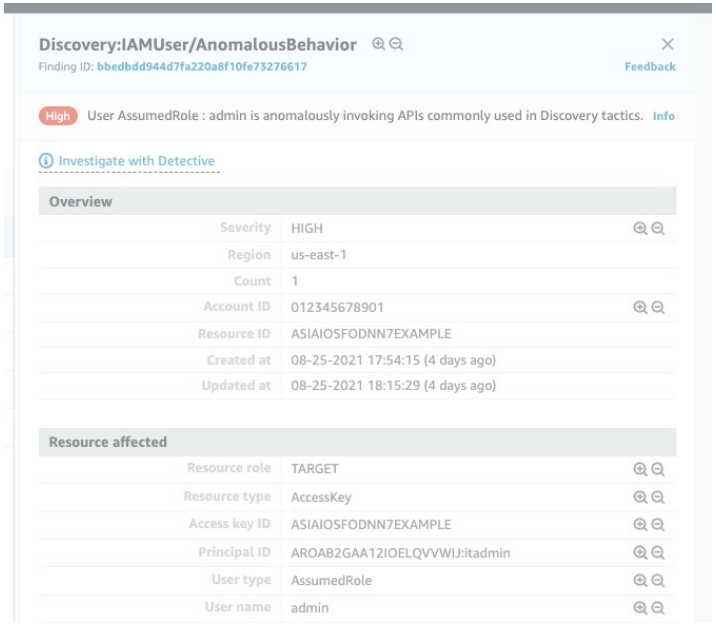


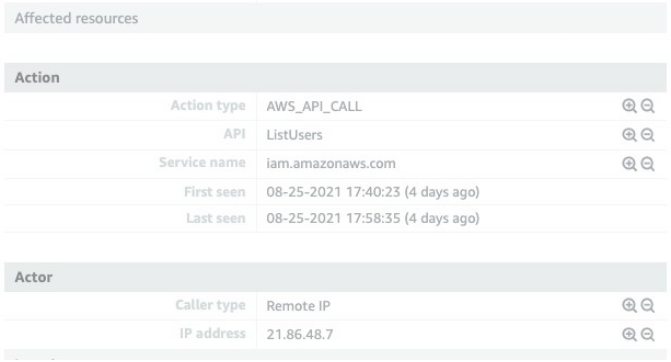
Hunters

Basic triage questions

1. **Who did it?**
2. **What did they do?**
3. From where?

We'll focus on these





Hunters

Disclaimer

There are simpler things than this methodology that can give 80% (Pareto) results.

We’re going for 99%.

Before we continue...

Prerequisites:

- CloudTrail logging enabled
- Logs accessible and queryable (Athena / DB / etc.)
- Best-practice CloudTrail configuration

Answering “who did it?”

We know that:

arn:aws:sts::222222222222:assumed-role/**admin/itadmin**

was used.

But - which person or system was using this identity?

This is non-trivial to answer accurately.

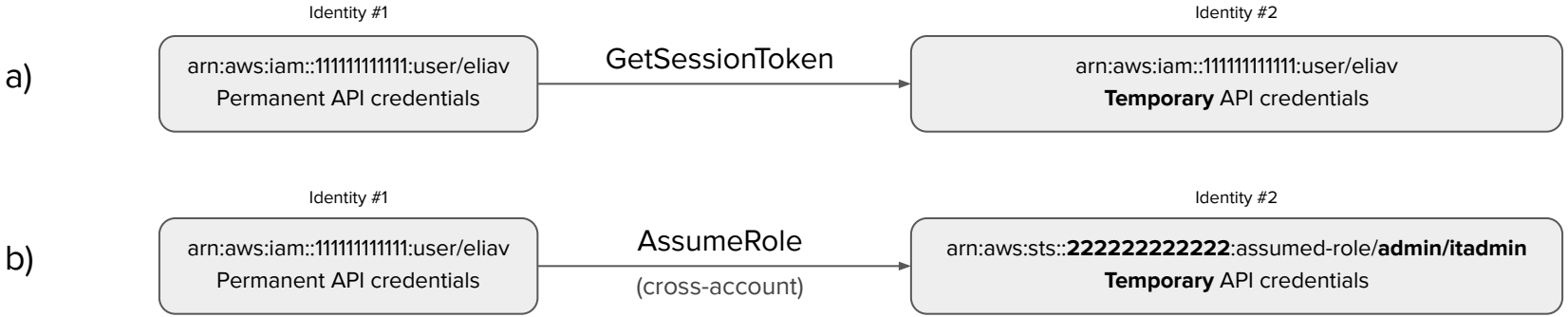
AWS Identity Chains

There are several API calls that pivot between identities.

Some pivot between AWS accounts, and/or change the identity ARN.

They all generate temporary credentials.

Examples:

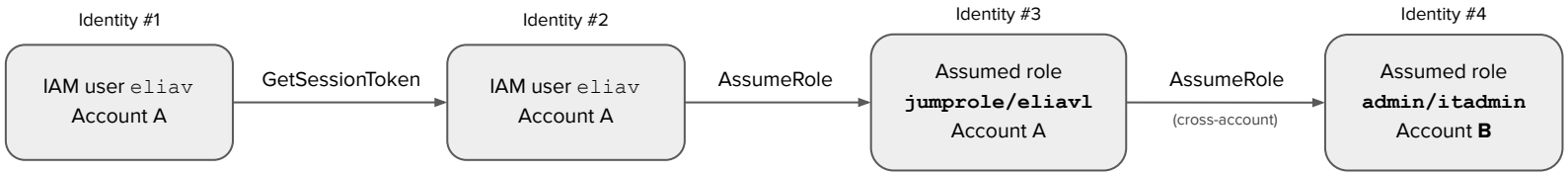


Hunters

AWS Identity Chains - more complex

GuardDuty gave us this

c)



By following the chain backwards, now we know “who did it”!

Hunters

Next question - “what did they do?”

- We have all ARNs related to the activity
- So, shall we just search for all events with the ARNs?
- …

Hunters

ARNs aren’t unique; access keys are

Multiple people/services can use “arn:aws:sts::222222222222:assumed-role/admin/itadmin” simultaneously.

Searching for all events with this ARN can lead to results from different “sessions”.

So, what are 100% unique*? API access keys!

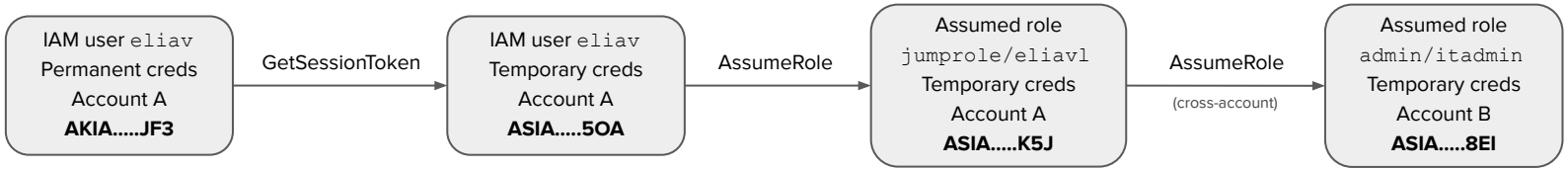
Following the access keys

With API calls, easy - just follow the identity chain.

Hunters

Following the access keys

With API calls, easy - just follow the identity chain.



However, web sessions are… a bit different.

- Web console only requires username and password
- All AWS API calls require API access key + secret key
- So… how does the web console *work*?

Web console internals

- User logs in to web with username and password
- Behind the scenes, temporary API credentials are dynamically generated
- Web console makes API calls on your behalf and displays results
- Different access key generated per region

How to find these API access keys?

Web console internals

- User logs in to web with username and password
- Behind the scenes, temporary credentials ar
- Web console uses these for making API

reg

or SwitchRole / RenewRole

```
"eventName": "ConsoleLogin",
"eventTime": "2021-08-27T19:42:56Z",
"userIdentity": {
    "arn": "arn:aws:iam::111111111111:user/eliav",
     ...
}
```

Browsing the EC2 service page

```
"eventName": "DescribeVolumeStatus",
"eventName": "DescribeAvailabilityZones",
"eventName": "DescribeInstances",
"eventTime": "2021-08-27T19:43:02Z",
"userIdentity": {
    "accessKeyId": "ASIAEXAMPLE01234567",
    "arn": "arn:aws:iam::111111111111:user/eliav",
    "sessionContext": {
        "attributes": {
            "creationDate": "2019-01-21T19:42:56Z",
        },
    },
}
```


Hunters

Web console internals

- User logs in to web with username and password
- Behind the scenes, temporary credentials are dynamically generated
- Web console uses these for making API calls on your behalf and displays results
- Different access key generated per region

How to find these API access keys?

OK, now we have all relevant access keys for the session!

Now what?

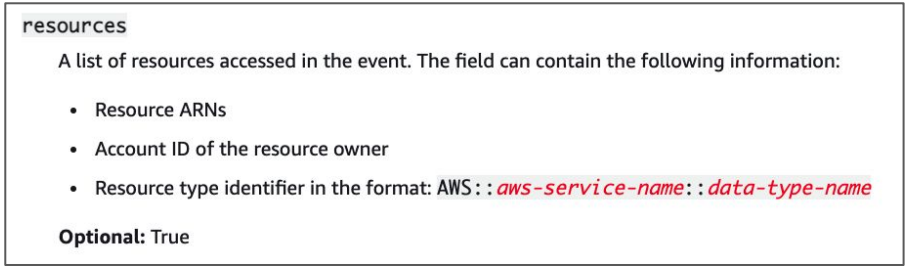
- Now we need to see which actions those access keys did during the session
- We can simply search for all CloudTrail events with these access keys!
- Well… *technically* true, but it's hard to consume:
  - Every single API call is logged separately
    - Simple web sessions often generate thousands of events
  - Logging is *not entirely* comprehensive
    - Varying levels of logging across services

Some CloudTrail quirks... #1

`resources` field

- Points to “on what was the action done”
- Critical for context
  - e.g. Prod DC vs. dev test machine

  Only being populated by ~10%-15% of the services :(
  - Non-populating services: IAM, EC2, Lambda, RDS, Secrets Manager, etc...
- We use this field when populated



`resources`

A list of resources accessed in the event. The field can contain the following information:

- Resource ARNs
- Account ID of the resource owner
- Resource type identifier in the format: `AWS::aws-service-name::data-type-name`

Optional: True

Hunters

Some CloudTrail quirks... #2

`readOnly` field

- Indicates if API call changed resources or not
- Helps zooming in on the sensitive/interesting actions
- ~~Isn't always populated :(~~
  - Historically, only ~20% populated
  - During Q4/20, 20% -> 100% population!
- Isn't necessarily a good indicator - these are read-only:
  - STS AssumeRole, GetSessionToken
  - S3 GetObject
  - So, we maintain a list of "sensitive read-only" events

`readOnly`

Identifies whether this operation is a read-only operation. This can be one of the following values:

- `true` – The operation is read-only (for example, `DescribeTrails`).
- `false` – The operation is write-only (for example, `DeleteTrail`).

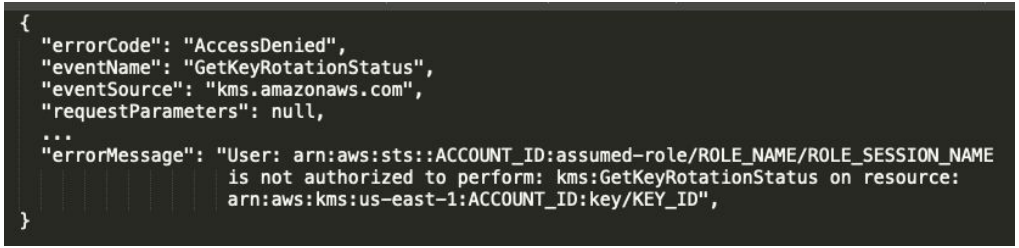
Optional: True

Hunters

Some CloudTrail quirks... #3

`requestParameters` field - for target resource

- When `resources` unpopulated, we can use requestParameters
  - Each API targeting a resource has some request parameter
- Unfortunately, the parameter names aren't standardized
  - For example, AttachRolePolicy receives `policyArn` but UpdatePolicy receives `policyId`
- Also, when API call fails with IAM AccessDenied, requestParameters is empty :(
  - However, errorMessage is not :)
- Generic parameter - filterSet:items
  - (in "resource-searching" API calls)



Hunters

Some CloudTrail quirks... #3

`requestParameters` field - for target resource

- When `resources` unpopulated, we can use requestParameters
  - Each API targeting a resource has some request parameter
- Unfortunately, the parameter names aren't standardized
  - For example, AttachRolePolicy receives `policyArn` but UpdatePolicy receives `policyId`
- Also, when API call fails with IAM AccessDenied, requestParameters is empty :(
  - However, errorMessage is not :)
- Generic parameter - filterSet:items
  - (in "resource-searching" API calls)


```
{
  "eventName": "DescribeInstances",
  "eventSource": "ec2.amazonaws.com",
  "requestParameters": {
    "filterSet": {"items":
      [{"name": "tag:Name", "valueSet": {"items": [{"value": "web-app-001-prod"}]}}]}
  }
}
```

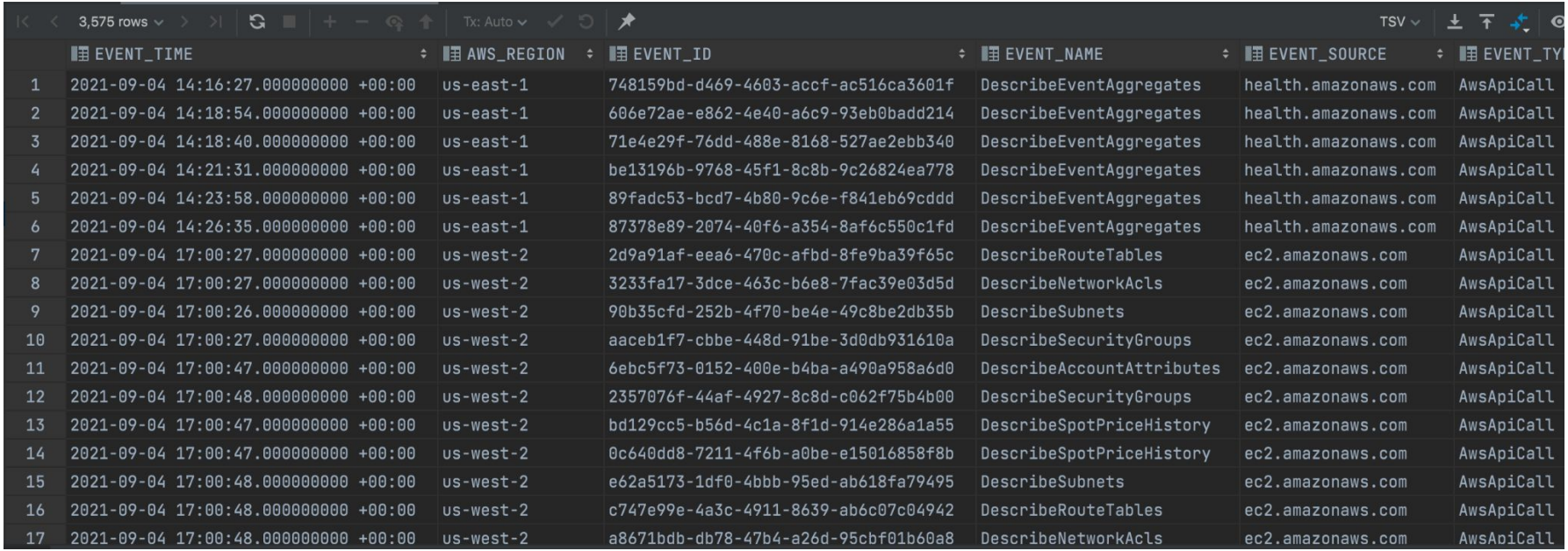

Hunters

So, best effort

To summarize all events in a consumable way, we group by API call, and:

- Count number of total calls
- Count successful / failed calls
- Extract a sample resources JSON (if populated)
- Extract target resource from IAM access denied error messages
- Extract a sample requestParameters JSON (for non-read-only events)
- Extract a sample requestParameters filter (for read-only events)
- Extract a sample error message

From this



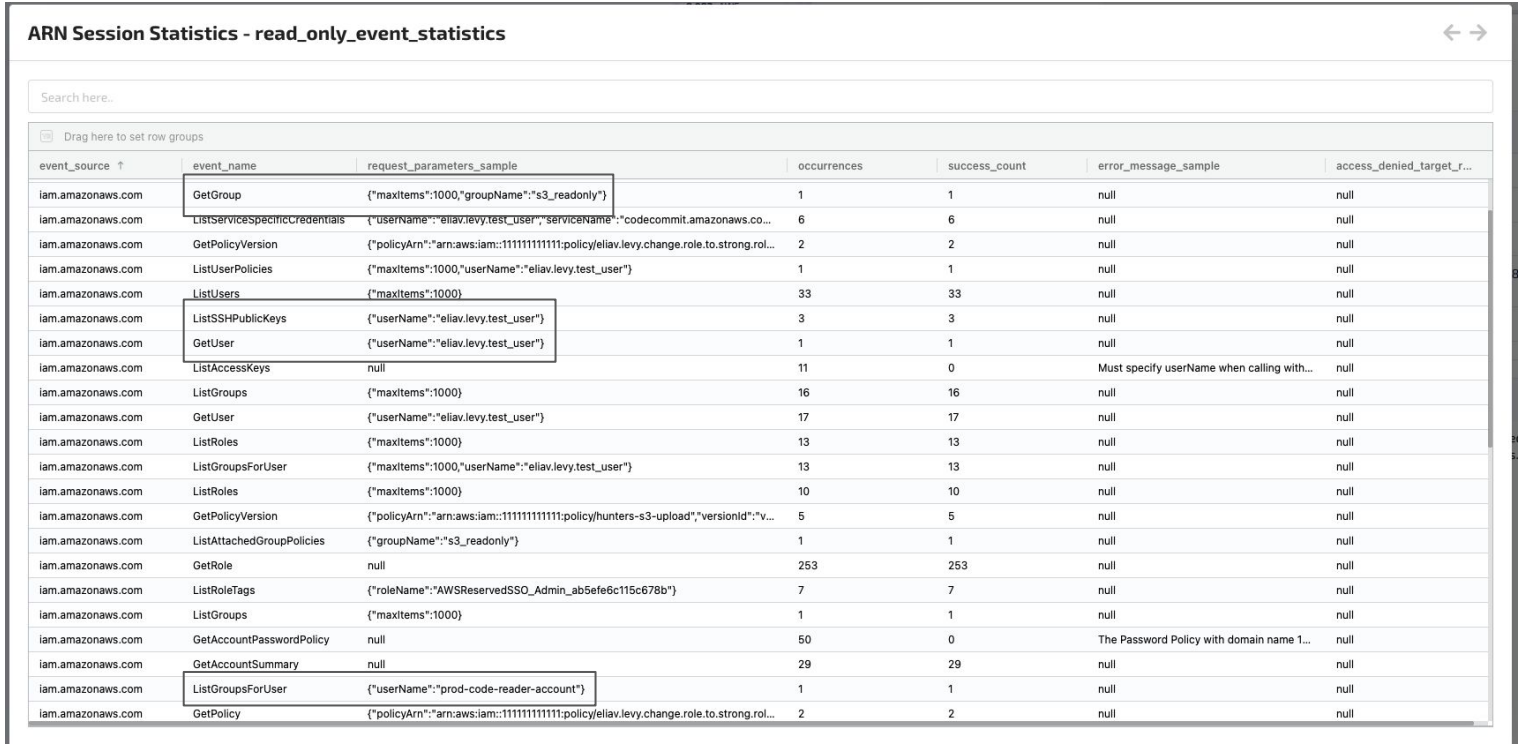
3,575 rows | Tx: Auto | TSV

	EVENT_TIME	AWS_REGION	EVENT_ID	EVENT_NAME	EVENT_SOURCE	EVENT_TY
1	2021-09-04 14:16:27.000000000 +00:00	us-east-1	748159bd-d469-4603-accf-ac516ca3601f	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
2	2021-09-04 14:18:54.000000000 +00:00	us-east-1	606e72ae-e862-4e40-a6c9-93eb0badd214	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
3	2021-09-04 14:18:40.000000000 +00:00	us-east-1	71e4e29f-76dd-488e-8168-527ae2ebb340	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
4	2021-09-04 14:21:31.000000000 +00:00	us-east-1	be13196b-9768-45f1-8c8b-9c26824ea778	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
5	2021-09-04 14:23:58.000000000 +00:00	us-east-1	89fadc53-bcd7-4b80-9c6e-f841eb69cddd	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
6	2021-09-04 14:26:35.000000000 +00:00	us-east-1	87378e89-2074-40f6-a354-8af6c550c1fd	DescribeEventAggregates	health.amazonaws.com	AwsApiCall
7	2021-09-04 17:00:27.000000000 +00:00	us-west-2	2d9a91af-eea6-470c-afbd-8fe9ba39f65c	DescribeRouteTables	ec2.amazonaws.com	AwsApiCall
8	2021-09-04 17:00:27.000000000 +00:00	us-west-2	3233fa17-3dce-463c-b6e8-7fac39e03d5d	DescribeNetworkAcls	ec2.amazonaws.com	AwsApiCall
9	2021-09-04 17:00:26.000000000 +00:00	us-west-2	90b35cfd-252b-4f70-be4e-49c8be2db35b	DescribeSubnets	ec2.amazonaws.com	AwsApiCall
10	2021-09-04 17:00:27.000000000 +00:00	us-west-2	aaceb1f7-cbbe-448d-91be-3d0db931610a	DescribeSecurityGroups	ec2.amazonaws.com	AwsApiCall
11	2021-09-04 17:00:47.000000000 +00:00	us-west-2	6ebc5f73-0152-400e-b4ba-a490a958a6d0	DescribeAccountAttributes	ec2.amazonaws.com	AwsApiCall
12	2021-09-04 17:00:48.000000000 +00:00	us-west-2	2357076f-44af-4927-8c8d-c062f75b4b00	DescribeSecurityGroups	ec2.amazonaws.com	AwsApiCall
13	2021-09-04 17:00:47.000000000 +00:00	us-west-2	bd129cc5-b56d-4c1a-8f1d-914e286a1a55	DescribeSpotPriceHistory	ec2.amazonaws.com	AwsApiCall
14	2021-09-04 17:00:47.000000000 +00:00	us-west-2	0c640dd8-7211-4f6b-a0be-e15016858f8b	DescribeSpotPriceHistory	ec2.amazonaws.com	AwsApiCall
15	2021-09-04 17:00:48.000000000 +00:00	us-west-2	e62a5173-1df0-4bbb-95ed-ab618fa79495	DescribeSubnets	ec2.amazonaws.com	AwsApiCall
16	2021-09-04 17:00:48.000000000 +00:00	us-west-2	c747e99e-4a3c-4911-8639-ab6c07c04942	DescribeRouteTables	ec2.amazonaws.com	AwsApiCall
17	2021-09-04 17:00:48.000000000 +00:00	us-west-2	a8671bdb-db78-47b4-a26d-95cbf01b60a8	DescribeNetworkAcls	ec2.amazonaws.com	AwsApiCall

(... thousands more rows)

Hunters

To this



ARN Session Statistics - read_only_event_statistics

Search here..

Drag here to set row groups

event_source	event_name	request_parameters_sample	occurrences	success_count	error_message_sample	access_denied_target_r...
iam.amazonaws.com	GetGroup	{"maxItems":1000,"groupName":"s3_readonly"}	1	1	null	null
iam.amazonaws.com	ListServiceSpecificCredentials	{"userName":"eliav.levy.test_user","serviceName":"codecommit.amazonaws.co...	6	6	null	null
iam.amazonaws.com	GetPolicyVersion	{"policyArn":"arn:aws:iam::111111111111:policy/eliav.levy.change.role.to.strong.rol...	2	2	null	null
iam.amazonaws.com	ListUserPolicies	{"maxItems":1000,"userName":"eliav.levy.test_user"}	1	1	null	null
iam.amazonaws.com	ListUsers	{"maxItems":1000}	33	33	null	null
iam.amazonaws.com	ListSSHPublicKeys	{"userName":"eliav.levy.test_user"}	3	3	null	null
iam.amazonaws.com	GetUser	{"userName":"eliav.levy.test_user"}	1	1	null	null
iam.amazonaws.com	ListAccessKeys	null	11	0	Must specify userName when calling with...	null
iam.amazonaws.com	ListGroups	{"maxItems":1000}	16	16	null	null
iam.amazonaws.com	GetUser	{"userName":"eliav.levy.test_user"}	17	17	null	null
iam.amazonaws.com	ListRoles	{"maxItems":1000}	13	13	null	null
iam.amazonaws.com	ListGroupsForUser	{"maxItems":1000,"userName":"eliav.levy.test_user"}	13	13	null	null
iam.amazonaws.com	ListRoles	{"maxItems":1000}	10	10	null	null
iam.amazonaws.com	GetPolicyVersion	{"policyArn":"arn:aws:iam::111111111111:policy/hunters-s3-upload","versionId":"v...	5	5	null	null
iam.amazonaws.com	ListAttachedGroupPolicies	{"groupName":"s3_readonly"}	1	1	null	null
iam.amazonaws.com	GetRole	null	253	253	null	null
iam.amazonaws.com	ListRoleTags	{"roleName":"AWSReservedSSO_Admin_ab5efe6c115c678b"}	7	7	null	null
iam.amazonaws.com	ListGroups	{"maxItems":1000}	1	1	null	null
iam.amazonaws.com	GetAccountPasswordPolicy	null	50	0	The Password Policy with domain name 1...	null
iam.amazonaws.com	GetAccountSummary	null	29	29	null	null
iam.amazonaws.com	ListGroupsForUser	{"userName":"prod-code-reader-account"}	1	1	null	null
iam.amazonaws.com	GetPolicy	{"policyArn":"arn:aws:iam::111111111111:policy/eliav.levy.change.role.to.strong.rol...	2	2	null	null

Tens of rows, summarized

Hunters

To this

ARN Session Statistics - not_read_only_event_statistics

Search here..

Drag here to set row groups

event_source	event_name	request_parameters_sample	access_denied_target_resource_sam...	error_message_sample	occurrences	success_co
iam.amazonaws.com	DetachUserPolicy	{"userName":"eliav.levy.test_user","policyArn":"arn:aws:iam::111111111111:policy/eliav.levy.second.p...	null	null	1	1
iam.amazonaws.com	AttachUserPolicy	{"userName":"eliav.levy.test_user","policyArn":"arn:aws:iam::aws:policy/AmazonS3FullAccess"}	null	null	3	3
iam.amazonaws.com	GenerateServiceLastAc...	{"arn":"arn:aws:iam::111111111111:role/aws-reserved/sso.amazonaws.com/AWSReservedSSO_Admi...	null	null	3	3
iam.amazonaws.com	CreateRole	{"tags":[],"roleName":"eliav.levy.test.strong.user.iam.ec2.s3","description":"Strong iam, ec2 and s...	null	null	2	1
iam.amazonaws.com	CreatePolicyVersion	{"policyArn":"arn:aws:iam::111111111111:policy/eliav.levy.change.role.to.strong.role.policy","setAsDe...	null	null	3	3
iam.amazonaws.com	CreatePolicy	{"tags":[],"policyName":"eliav.levy.change.role.to.strong.role.policy","policyDocument":"{\n \"Versi...	null	null	3	3
iam.amazonaws.com	CreateInstanceProfile	{"instanceProfileName":"eliav.levy.test.strong.user.iam.ec2.s3"}	null	null	1	1
iam.amazonaws.com	CreateUser	{"tags":[],"userName":"eliav.levy.test_user"}	null	null	1	1
iam.amazonaws.com	DeletePolicy	{"policyArn":"arn:aws:iam::111111111111:policy/eliav.levy.second.poilcy.try.strong.user"}	null	null	1	1
iam.amazonaws.com	AttachRolePolicy	{"roleName":"eliav.levy.test.strong.user.iam.ec2.s3","policyArn":"arn:aws:iam::aws:policy/AmazonE...	null	null	2	2
iam.amazonaws.com	CreateAccessKey	{"userName":"eliav.levy.test_user"}	null	null	1	1
iam.amazonaws.com	AddRoleToInstanceProfile	{"roleName":"eliav.levy.test.strong.user.iam.ec2.s3","instanceProfileName":"eliav.levy.test.strong.u...	null	null	1	1
signin.amazonaws.com	SwitchRole	null	null	switchrole.error.invalidparams	10	0
signin.amazonaws.com	ConsoleLogin	null	null	null	6	6
sts.amazonaws.com	AssumeRole	null	arn:aws:iam::111111111111:role/eliav.levy.test...	null	10	0
sts.amazonaws.com	AssumeRole	null	eliav.levy.test.strong.user.iam.ec2.s3	null	1	0

Hunters

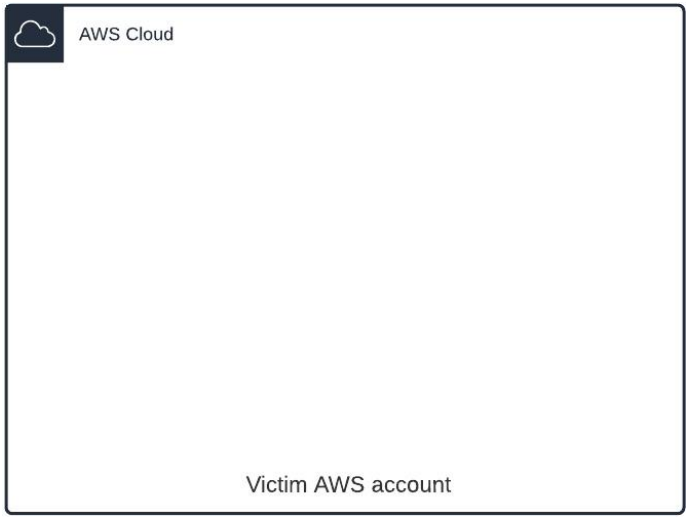
Moving on - more CloudTrail quirks

Fun quirk #1 - CloudTrail source IP spoofing

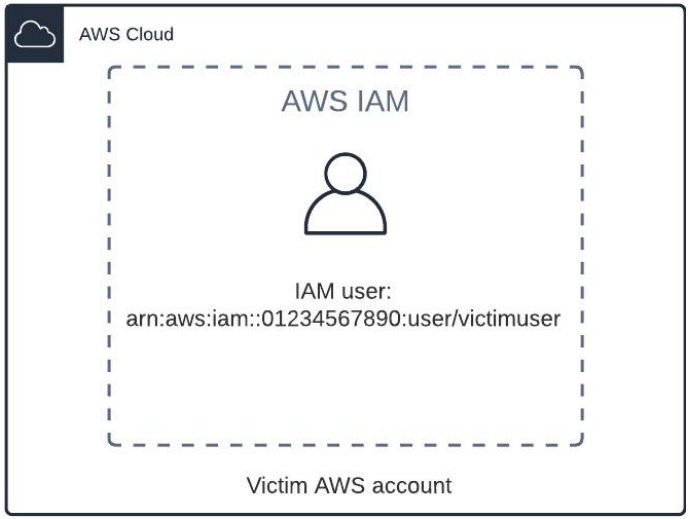
When using compromised AWS API credentials, an attacker can spoof the source IP that appears in the victim's CloudTrail logs.

Technique Overview

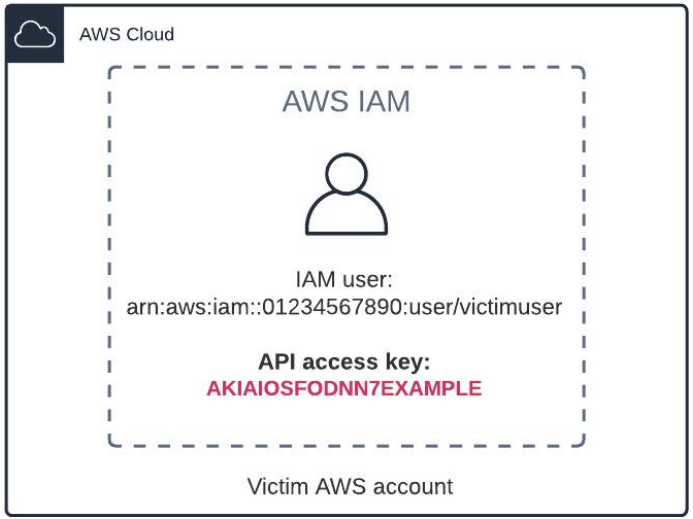
Technique Overview



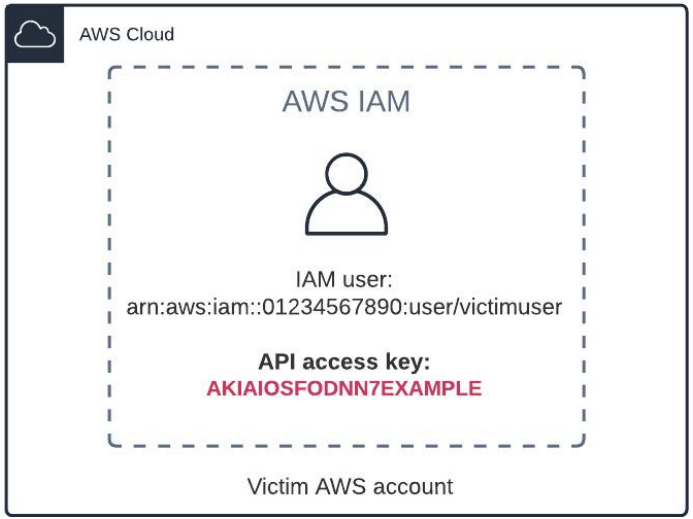
Technique Overview



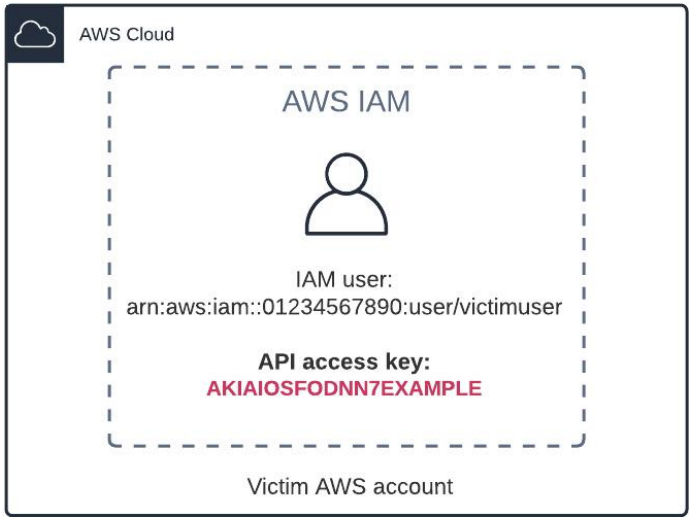
Technique Overview



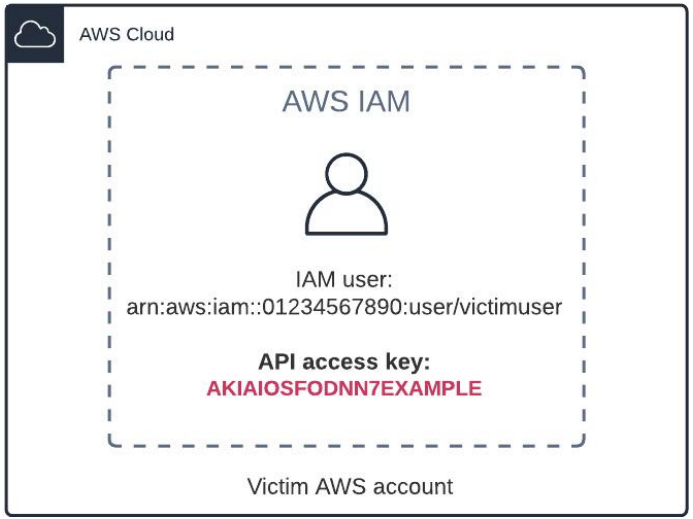
Technique Overview

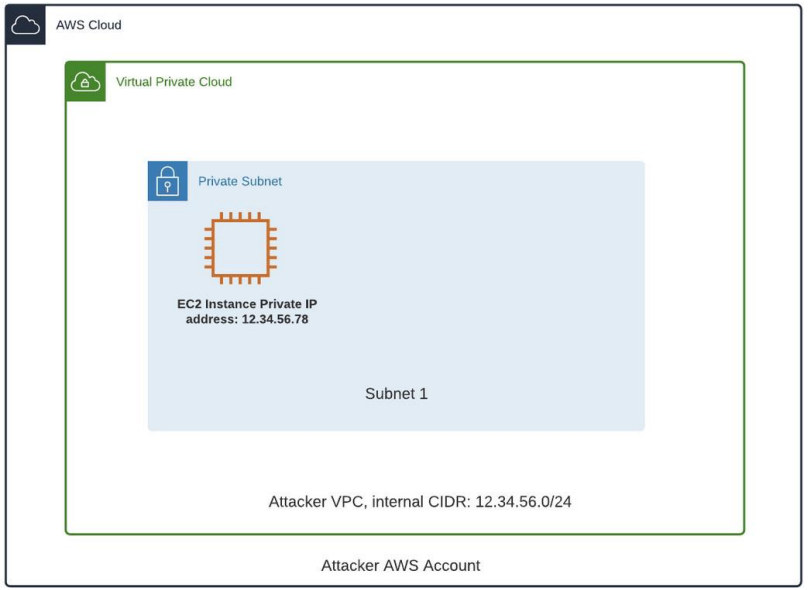


Technique Overview

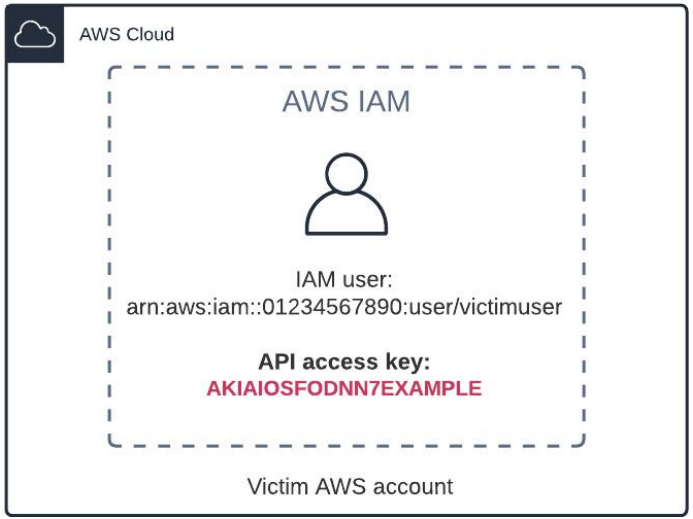


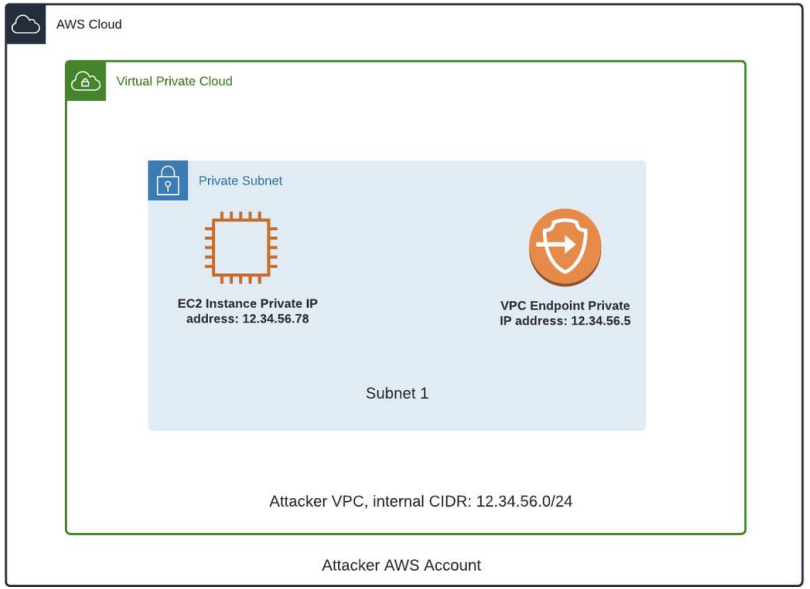
Technique Overview





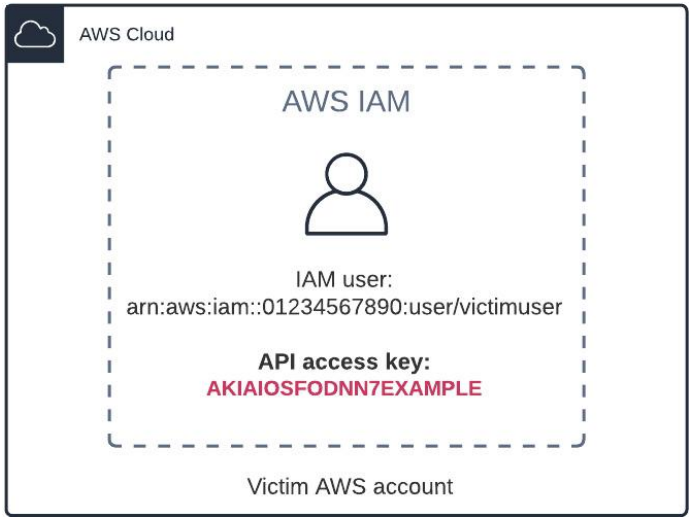
Technique Overview

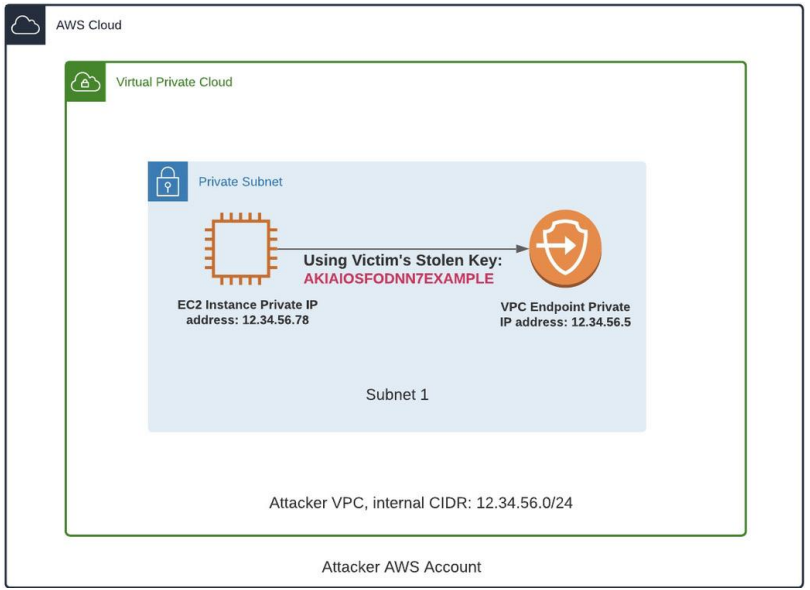




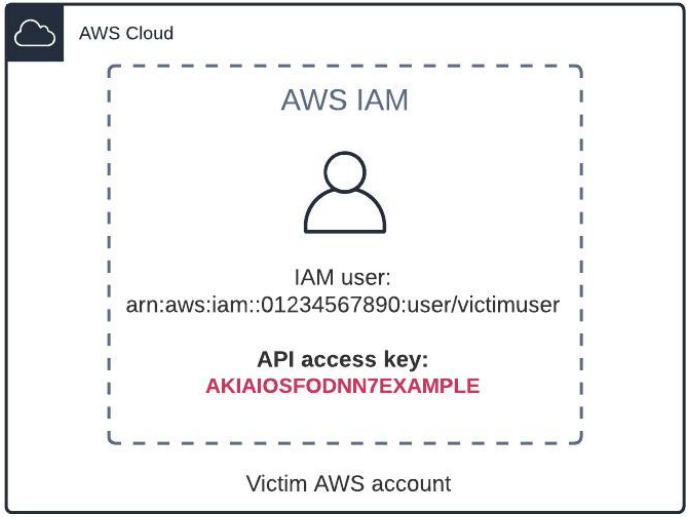
Hunters

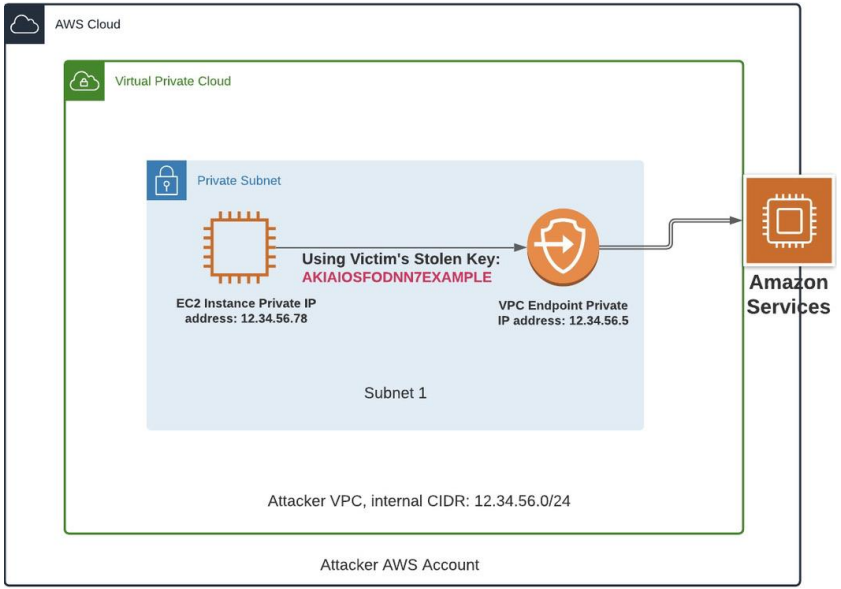
Technique Overview





Technique Overview

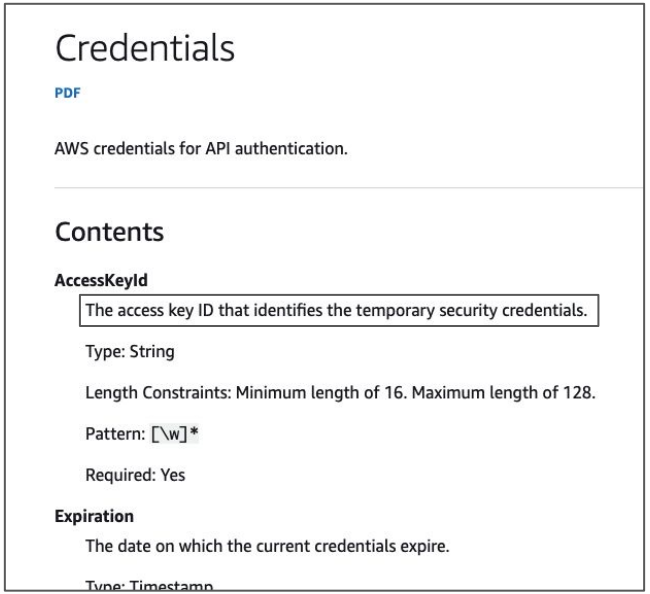




Technique Overview

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…



Credentials

PDF

AWS credentials for API authentication.

Contents

AccessKeyId

The access key ID that identifies the temporary security credentials.

Type: String

Length Constraints: Minimum length of 16. Maximum length of 128.

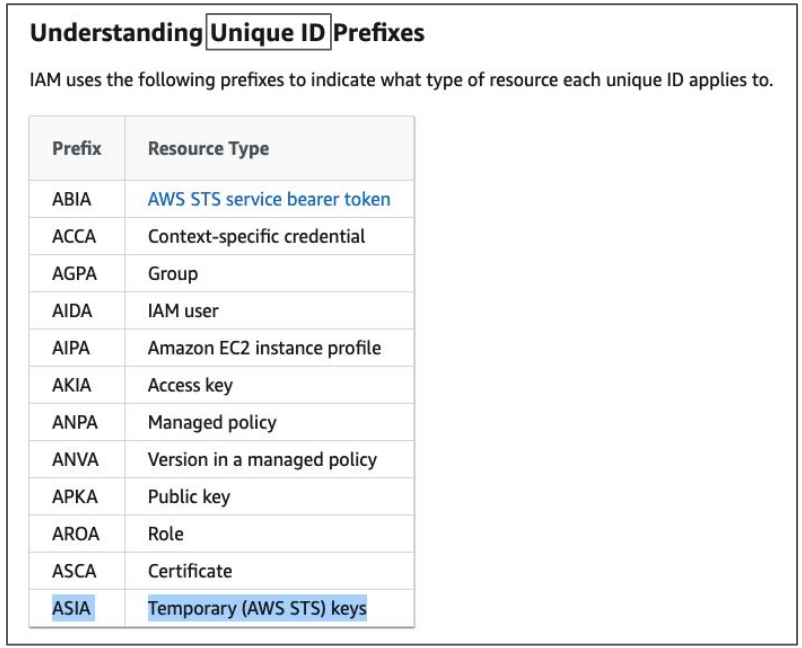
Pattern: `[\w]*`

Required: Yes

Expiration

The date on which the current credentials expire.

Type: Timestamp



Understanding Unique ID Prefixes

IAM uses the following prefixes to indicate what type of resource each unique ID applies to.

Prefix	Resource Type
ABIA	AWS STS service bearer token
ACCA	Context-specific credential
AGPA	Group
AIDA	IAM user
AIPA	Amazon EC2 instance profile
AKIA	Access key
ANPA	Managed policy
ANVA	Version in a managed policy
APKA	Public key
AROA	Role
ASCA	Certificate
ASIA	Temporary (AWS STS) keys

Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique...
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                                    _6fd7f06b",
      "assumedRoleId": "AROA                             _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIA           TEASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "
                    9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                                      _e6880947",
      "assumedRoleId": "AROA                             _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIA           TEASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "
                    msA4Zq1iM="
    }
  }
}
```


Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique...
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                    _6fd7f06b",
      "assumedRoleId": "AROA                  _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIA       TEASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "
                    9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                    _e6880947",
      "assumedRoleId": "AROA                  _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIA       TEASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "
                    msA4Zq1iM="
    }
  }
}
```


Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                                    _6fd7f06b",
      "assumedRoleId": "AROA                                   _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIA           TEASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "
                                                                                                  9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                                    _e6880947",
      "assumedRoleId": "AROA                                   _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIA           TEASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "
                                                                                                  msA4Zq1iM="
    }
  }
}
```


Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::  _6fd7f06b",
      "assumedRoleId": "AROA  _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIAX  EASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "  9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::  _e6880947",
      "assumedRoleId": "AROA  _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIAX  EASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "  msA4Zq1iM="
    }
  }
}
```

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique...
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                    _6fd7f06b",
      "assumedRoleId": "AROA                   _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIA         TEASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "
                 9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                    _e6880947",
      "assumedRoleId": "AROA                   _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIA         TEASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "
                 msA4Zq1iM="
    }
  }
}
```


Hunters

Fun quirk #2 - API access key recycling

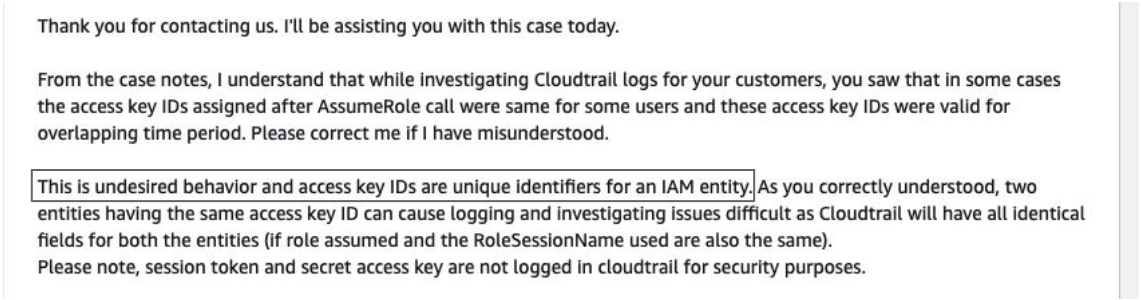
- API access keys are supposed to be unique...
- Or so I thought


```
{
  "awsRegion": "us-east-1",
  "eventID": "889087b1-852f-4871-b1bd-cc7bde9dff0c",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T19:36:15Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                d7f06b",
      "assumedRoleId": "AROA                _6fd7f06b"
    },
    "credentials": {
      "accessKeyId": "ASIA          TEASS",
      "expiration": "Aug 21, 2021 8:36:15 PM",
      "sessionToken":
       "
                    9UOs5x3qs="
    }
  }
}
```

```
  "awsRegion": "us-east-1",
  "eventID": "368084bd-cdab-4be9-95b2-d304c57dea25",
  "eventName": "AssumeRole",
  "eventSource": "sts.amazonaws.com",
  "eventTime": "2021-08-21T20:01:51Z",
  "responseElements": {
    "assumedRoleUser": {
      "arn": "arn:aws:sts::                880947",
      "assumedRoleId": "AROA                _e6880947"
    },
    "credentials": {
      "accessKeyId": "ASIA          TEASS",
      "expiration": "Aug 21, 2021 9:01:51 PM",
      "sessionToken":
       "
                    msA4Zq1iM="
    }
  }
}
```

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought
- Turns out they are recycled across identities



Thank you for contacting us. I'll be assisting you with this case today.

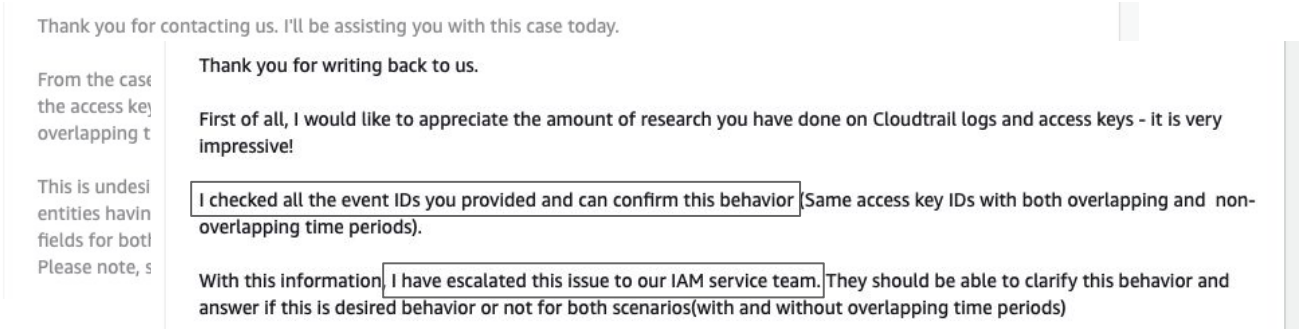
From the case notes, I understand that while investigating Cloudtrail logs for your customers, you saw that in some cases the access key IDs assigned after AssumeRole call were same for some users and these access key IDs were valid for overlapping time period. Please correct me if I have misunderstood.

This is undesired behavior and access key IDs are unique identifiers for an IAM entity. As you correctly understood, two entities having the same access key ID can cause logging and investigating issues difficult as Cloudtrail will have all identical fields for both the entities (if role assumed and the RoleSessionName used are also the same).
Please note, session token and secret access key are not logged in cloudtrail for security purposes.

Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought
- Turns out they are recycled across identities



Thank you for contacting us. I'll be assisting you with this case today.

From the case
the access key
overlapping t

This is undesi
entities havin
fields for both
Please note, s

Thank you for writing back to us.

First of all, I would like to appreciate the amount of research you have done on Cloudtrail logs and access keys - it is very impressive!

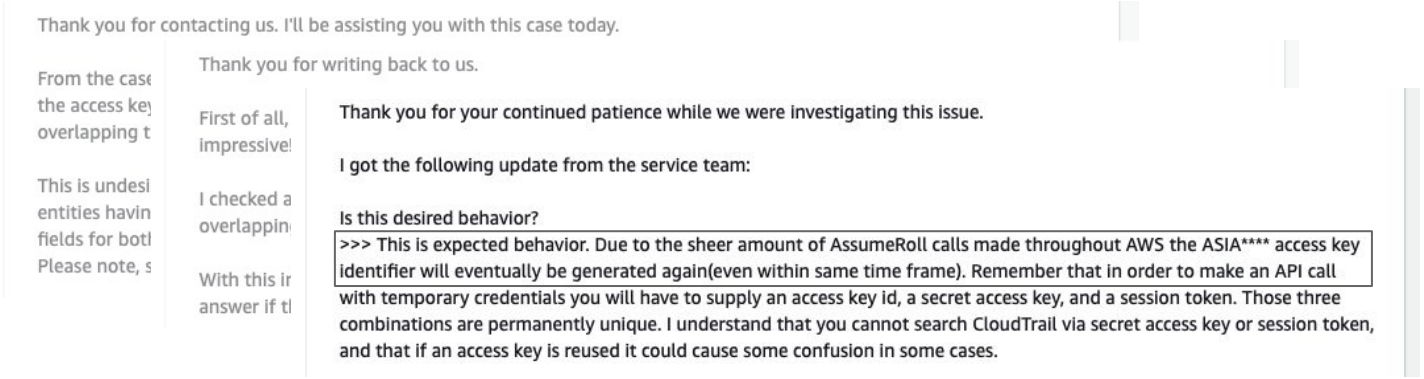
I checked all the event IDs you provided and can confirm this behavior (Same access key IDs with both overlapping and non-overlapping time periods).

With this information, I have escalated this issue to our IAM service team. They should be able to clarify this behavior and answer if this is desired behavior or not for both scenarios(with and without overlapping time periods)

Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought
- Turns out they are recycled across identities



Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought
- Turns out they are recycled across identities

Understanding Unique ID Prefixes

IAM uses the following prefixes to indicate what type of resource each unique ID applies to.

Prefix	Resource Type
ABIA	AWS STS service bearer token
ACCA	Context-specific credential
AGPA	Group
AIDA	IAM user
AIPA	Amazon EC2 instance profile
AKIA	Access key
ANPA	Managed policy
ANVA	Version in a managed policy
APKA	Public key
AROA	Role
ASCA	Certificate
ASIA	Temporary (AWS STS) keys

Understanding unique ID prefixes

IAM uses the following prefixes to indicate what type of resource each unique ID applies to.

Prefix	Resource type
ABIA	AWS STS service bearer token
ACCA	Context-specific credential
AGPA	User group
AIDA	IAM user
AIPA	Amazon EC2 instance profile
AKIA	Access key
ANPA	Managed policy
ANVA	Version in a managed policy
APKA	Public key
AROA	Role
ASCA	Certificate
ASIA	Temporary (AWS STS) access key IDs use this prefix but are unique only in combination with the secret access key and the session token.

Hunters

Fun quirk #2 - API access key recycling

- API access keys are supposed to be unique…
- Or so I thought
- Turns out they are recycled across identities
- Not a security risk
  - Happens only rarely
  - Only recycled in-account
  - No “cross-permissions” possible

What AWS can improve

- More comprehensive logging
- Better documentation
- Improve AWS detective

Thank you

Feedback appreciated!

bit.ly/fwdcs21-levy

/eliav-levy

eliav@hunters.ai

Hunters

Questions

Feedback appreciated!

bit.ly/fwdcs21-levy

/eliav-levy

eliav@hunters.ai

Hunters