WeirdAAL

https://github.com/carnal0wnage/weirdAAL

Run recon against all AWS services to enumerate access for a set of keys

python3 weirdAAL.py -m recon_all -t <name>

Pacu

AWS exploitation framework

https://github.com/RhinoSecurityLabs/pacu

Install Pacu

```
sudo apt-get install python3-pip
git clone https://github.com/RhinoSecurityLabs/pacu
cd pacu
sudo bash install.sh
```

Import AWS keys for a specific profile

```
import_keys <profile name>
```

Detect if keys are honey token keys

```
run iam__detect_honeytokens
```

Enumerate account information and permissions

```
run iam__enum_users_roles_policies_groups
run iam__enum_permissions
whoami
```

Check for privilege escalation

```
run iam__privesc_scan
```

Scanning Loops

Scan (sts check, scoutsuite, Prowler, and CloudFox) across multiple accounts

```
while read r; do echo $r; aws sts get-caller-identity--profile $r; done < accounts.txt
while read r; do echo $r; scout aws--profile $r; done < accounts.txt
while read r; do echo $r; prowler aws-q-p $r; done < accounts.txt
while read r; do echo $r; ./cloudfox aws all-checks-p $r; done < accounts.txt
```

Run aws_public_ips across all accounts https://github.com/arkadiyt/aws_public_ips

```
while read p; do
  echo $p
while read r; do
  echo $r
  AWS_PROFILE=$p AWS_REGION=$r aws_public_ips >> aws_public_ips.txt
done < regions.txt
done < accounts.txt
```

ScoutSuite

Multi-cloud security auditing tool

https://github.com/nccgroup/ScoutSuite

Install ScoutSuite

```
sudo apt-get install virtualenv
git clone https://github.com/nccgroup/ScoutSuite
cd ScoutSuite
virtualenv –p python3 venv
source venv/bin/activate
pip install –r requirements.txt
```

To run as root

```
sudo apt-get install virtualenv
sudo su
virtualenv-p python3 venv
source venv/bin/activate
pip install scoutsuite
```

Scan AWS environment with ScoutSuite

```
python scout.py aws --profile=<aws profile name>
```

or if installed...

scout aws --profile=<aws profile name>

jq queries to help with parsing many ScoutSuite reports

Sometimes you may need to work with multiple ScoutSuite files and report similar items across all of them. The ScoutSuite reports are in json format so the 'jq' tool can be used to parse through them easily. Here are a few short script examples for doing this. Run these from the directory where you output each of the ScoutSuite folders to.

AWS

### Find all ec2 ebs volumes unencrypted

```
for d in scoutsuite_results_aws-* ;do tail $d-n +2 | jq-r '.services.ec2.regions[].volumes[] | select(.Encrypted == false) | .arn' >> ec2-ebs-volume-not-encrypted.txt; done
```

### Find all ec2 ebs snapshots unencrypted

```
for d in scoutsuite_results_aws-* ;do tail $d-n +2 | jq-r '.services.ec2.regions[].snapshots[] | select(.encrypted == false) | .arn' >> ec2-ebs-snapshot-not-encrypted.txt; done
```

### Inline Role Policy Contains NotActions

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("Inline role Policy Allows \"NotActions\"")) | .items[]' | sed 's/\.inline_policies.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-inline-role-policy-contains-notactions.txt; done; done
```

### Inline Role Policy Allows iam:PassRole for All Resources

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("Inline role Policy Allows \"iam:PassRole\" For All Resources")) | .items[]' | sed 's/\.inline_policies.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-inline-role-policy-allows-iampassrole-for-all-resources.txt; done; done
```

### Inline Role Policy Allows sts:AssumeRole for All Resources

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Inline role Policy Allows \"sts:AssumeRole\" For All Resources")) | .items[]' | sed
's/\.inline_policies.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-inline-role-policy-allows-
stsassumerole-for-all-resources.txt; done; done
```

### Managed Policy Allows iam:PassRole for All Resources (without account numbers to get count)

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows \"iam:PassRole\" For All Resources")) | .items[]' | sed
's/\.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-
iampassrole-for-all-resources-NOACCOUNTNUMBERS.txt; done; done
```

### Managed Policy Allows iam:PassRole for All Resources (with account numbers)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-managed-policy-allows-
iampassrole-for-all-resources.txt;for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows \"iam:PassRole\" For All Resources")) | .items[]' | sed
's/\.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-
iampassrole-for-all-resources.txt; done; done
```

### Managed Policy Allows NotActions (without account numbers to get counts)

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows \"NotActions\"")) | .items[]' | sed
's/\.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-
notactions-NOACCOUNTNUMBERS.txt; done; done
```

### Managed Policy Allows NotActions (with account numbers)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-managed-policy-allows-
notactions.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description |
contains("Managed Policy Allows \"NotActions\"")) | .items[]' | sed 's/\.PolicyDocument.*//'); do tail $d-n
+2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-notactions.txt; done; done
```

### Managed Policy Allows sts:AssumeRole for All Resources (without account numbers)

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows \"sts:AssumeRole\" For All Resources")) | .items[]' | sed
's/\.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-
stsassumerole-for-all-resources-NOACCOUNTNUMBERS.txt; done; done
```

### Managed Policy Allows sts:AssumeRole for All Resources (with account numbers)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-managed-policy-allows-
stsassumerole-for-all-resources.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows \"sts:AssumeRole\" For All Resources")) | .items[]' | sed
's/\.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-
stsassumerole-for-all-resources.txt; done; done
```

### Managed Policy Allows All Actions (without account numbers)

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Managed Policy Allows All Actions")) | .items[]' | sed 's/\.PolicyDocument.*//');
do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-managed-policy-allows-allactions-
NOACCOUNTNUMBERS.txt; done; done
```

### Managed Policy Allows All Actions (with account numbers)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-managed-policy-allows-
allactions.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description |
contains("Managed Policy Allows All Actions")) | .items[]' | sed 's/\.PolicyDocument.*//'); do tail $d-n +2 |
jq-r ".services.$item | .arn" >> iam-managed-policy-allows-allactions.txt; done; done
```

### Cross-Account AssumeRole Policy Lacks External ID and MFA (without account numbers)

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select
(.description | contains("Cross-Account AssumeRole Policy Lacks External ID and MFA")) | .items[]' | sed
's/\.assume_role_policy.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-cross-account-policy-
lacks-external-id-and-mfa-NOACCOUNTNUMBERS.txt; done; done
```

### Cross-Account AssumeRole Policy Lacks External ID and MFA (with account numbers)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-cross-account-policy-lacks-external-id-and-mfa.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("Cross-Account AssumeRole Policy Lacks External ID and MFA")) | .items[]' | sed 's/\.assume_role_policy.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-cross-account-policy-lacks-external-id-and-mfa.txt; done; done
```

### Assume Role Policy Allows All Principals

```
for d in scoutsuite_results_aws-* ; do for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("AssumeRole Policy Allows All Principals")) | .items[]' | sed 's/\.assume_role_policy.PolicyDocument.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-assumerole-policy-allows-allprincipals-NOACCOUNTNUMBERS.txt; done; done
```

### Lack of Key Rotation for Active Days

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-lack-of-key-rotation-for-active-days.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("Lack of Key Rotation for (Active) Days")) | .items[]' | sed 's/\.AccessKeys.*//'); do tail $d-n +2 | jq -r ".services.$item | .arn" >> iam-lack-of-key-rotation-for-active-days.txt; done; done
```

### Lack of Key Rotation for Active Days (with create date)

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-lack-of-key-rotation-for-active-days-WITHCREATEDATE.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("Lack of Key Rotation for (Active) Days")) | .items[]' | sed 's/\.AccessKeys.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn,.AccessKeys[].CreateDate" >> iam-lack-of-key-rotation-for-active-days-WITHCREATEDATE.txt; done; done
```

### Users Without MFA

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-user-without-mfa.txt; for item in $(tail $d-n +2 | jq-r '.services.iam.findings[] | select (.description | contains("User without MFA")) | .items[]' | sed 's/\.mfa_enabled.*//'); do tail $d-n +2 | jq-r ".services.$item | .arn" >> iam-user-without-mfa.txt; done; done
```

### Password Policy

```
for d in scoutsuite_results_aws-* ; do tail $d-n +2 | jq-r '.account_id' >> iam-password-policy.txt; tail $d-n +2 | jq-r '.services.iam.password_policy' >> iam-password-policy.txt; done
```

### CloudTrail Service Not Configured

```
for d in scoutsuite_results_aws-* ; do echo " " >> cloudtrail-service-not-configured.txt; tail $d-n +2 | jq-r '.account_id' >> cloudtrail-service-not-configured.txt; tail $d-n +2 | jq-r '.services.cloudtrail.findings[] | select (.description | contains("Service Not Configured")) | .items[]' | sed 's/\.NotConfigured*//' >> cloudtrail-service-not-configured.txt; done
```

---------- OLD QUERIES THAT NEED UPDATING ----------

### Find All Lambda Environment Variables

```
for d in */ ; do
	tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.services.awslambda.regions[].functions[] | select (.env_variables != []) | .arn, .env_variables' >> lambda-all-environment-variables.txt
done
```

### Find World Listable S3 Buckets

```
for d in */ ; do
	tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.account_id, .services.s3.findings."s3-bucket-AuthenticatedUsers-read".items[]'  >> s3-buckets-world-listable.txt
done
```

### Find All EC2 User Data

```
for d in */ ; do
	tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.services.ec2.regions[].vpcs[].instances[] | select (.user_data != null) | .arn, .user_data'  >> ec2-instance-all-user-data.txt
done
```

### Find EC2 Security Groups That Whitelist AWS CIDRs

```
for d in */ ; do
	tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.account_id' >> ec2-security-group-whitelists-aws-cidrs.txt
	tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.services.ec2.findings."ec2-security-group-whitelists-aws".items'  >> ec2-security-group-whitelists-aws-cidrs.txt
done
```

### Find EC2 EBS Public AMIs

```
for d in */ ; do
        tail $d/scoutsuite-results/scoutsuite_results*.js-n +2 | jq '.services.ec2.regions[].images[] | select
(.Public == true) | .arn' >> ec2-public-amis.txt
done
```

🦊 CloudFox 🦊

CloudFox helps you gain situational awareness in unfamiliar cloud environments. It's an open source command line tool created to help penetration testers and other offensive security professionals find exploitable attack paths in cloud infrastructure.

CloudFox helps you answer the following common questions (and many more):

- What regions is this AWS account using and roughly how many resources are in the account?
- What secrets are lurking in EC2 userdata or service specific environment variables?
- What workloads have administrative permissions attached?
- What actions/permissions does this [principal] have?
- What role trusts are overly permissive or allow cross-account assumption?
- What endpoints/hostnames/IPs can I attack from an external starting point (public internet)?
- What endpoints/hostnames/IPs can I attack from an internal starting point (assumed breach within the VPC)?
- What filesystems can I potentially mount from a compromised resource inside the VPC?

Install

Option 1: Download the latest binary release for your platform.

Option 2: If you use homebrew: brew install cloudfox
Option 3: Install Go, use go install github.com/BishopFox/cloudfox@latest to install from the remote source
Option 4: Developer mode:

Install Go, clone the CloudFox repository and compile from source

```
# git clone https://github.com/BishopFox/cloudfox.git
...omitted for brevity...
# cd ./cloudfox
# Make any changes necessary
# go build .
# ./cloudfox
```

Prerequisites

AWS

- AWS CLI installed
- Supports AWS profiles, AWS environment variables, or metadata retrieval (on an ec2 instance)
  - To run commands on multiple profiles at once, you can specify the path to a file with a list of profile names separated by a new line using the -l flag or pass all stored profiles with the -a flag.
- A principal with one recommended policies attached (described below)
- Recommended attached policies: SecurityAudit + CloudFox custom policy

Additional policy notes (as of 09/2022):

Policy	Notes
CloudFox custom policy	Has a complete list of every permission cloudfox uses and nothing else
arn:aws:iam::aws:policy/SecurityAudit	Covers most cloudfox checks but is missing newer services or permissions like apprunner:*, grafana:*, lambda:GetFunctionURL, lightsail:GetContainerServices
arn:aws:iam::aws:policy/job-function/ViewOnlyAccess	Covers most cloudfox checks but is missing newer services or permissions like AppRunner:*, grafana:*, lambda:GetFunctionURL, lightsail:GetContainerServices- and is also missing iam:SimulatePrincipalPolicy.

Policy	Notes
arn:aws:iam::aws:policy/ReadOnlyAccess	Only missing AppRunner, but also grants things like "s3:Get*" which can be overly permissive.
arn:aws:iam::aws:policy/AdministratorAccess	This will work just fine with CloudFox, but if you were handed this level of access as a penetration tester, that should probably be a finding in itself :)

AWS Commands

Provider	Command Name	Description
AWS	all-checks	Run all of the other commands using reasonable defaults. You'll still want to check out the non-default options of each command, but this is a great place to start.
AWS	access-keys	Lists active access keys for all users. Useful for cross referencing a key you found with which in-scope account it belongs to.
AWS	buckets	Lists the buckets in the account and gives you handy commands for inspecting them further.
AWS	cloudformation	Lists the cloudformation stacks in the account. Generates loot file with stack details, stack parameters, and stack output- look for secrets.
AWS	ecr	List the most recently pushed image URI from all repositories. Use the loot file to pull selected images down with docker/nerdctl for inspection.

Provider	Command Name	Description
AWS	ecs-tasks	List all ecs tasks. This returns a list of ecs tasks and associated cluster, task definition, container instance, launch type, and associated IAM principal.
AWS	eks	List all EKS clusters, see if they expose their endpoint publicly, and check the associated IAM roles attached to reach cluster or node group. Generates a loot file with the aws eks udpate-kubeconfig command needed to connect to each cluster.
AWS	elastic-network-interfaces	List all eni information. This returns a list of eni ID, type, external IP, private IP, VPCID, attached instance and a description.
AWS	endpoints	Enumerates endpoints from various services. Scan these endpoints from both an internal and external position to look for things that don't require authentication, are misconfigured, etc.
AWS	env-vars	Grabs the environment variables from services that have them (App Runner, ECS, Lambda, Lightsail containers, Sagemaker are supported. If you find a sensitive secret, use cloudfox iam-simulator AND pmapper to see who has access to them.
AWS	filesystems	Enumerate the EFS and FSx filesystems that you might be able to mount without creds (if you have the right network access). For example, this is useful when you have ec:RunInstance but not iam:PassRole.
AWS	iam-simulator	Like pmapper, but uses the IAM policy simulator. It uses AWS's evaluation logic, but notably, it doesn't consider transitive access via privesc, which is why you should also always also use pmapper.

Provider	Command Name	Description
AWS	instances	Enumerates useful information for EC2 Instances in all regions like name, public/private IPs, and instance profiles. Generates loot files you can feed to nmap and other tools for service enumeration.
AWS	inventory	Gain a rough understanding of size of the account and preferred regions.
AWS	lambda	Lists the lambda functions in the account, including which one's have admin roles attached. Also gives you handy commands for downloading each function.
AWS	network-ports	Enumerates AWS services that are potentially exposing a network service. The security groups and the network ACLs are parsed for each resource to determine what ports are potentially exposed.
AWS	outbound-assumed-roles	List the roles that have been assumed by principals in this account. This is an excellent way to find outbound attack paths that lead into other accounts.
AWS	permissions	Enumerates IAM permissions associated with all users and roles. Grep this output to figure out what permissions a particular principal has rather than logging into the AWS console and painstakingly expanding each policy attached to the principal you are investigating.
AWS	principals	Enumerates IAM users and Roles so you have the data at your fingertips.
AWS	pmapper	Looks for pmapper data stored on the local filesystem, in the locations defined here. If pmapper data has been found (you already ran pmapper graph create), then this command will use this data to build a graph in cloudfox memory let you know who can privesc to admin.

Provider	Command Name	Description
AWS	principals	Enumerates IAM users and Roles so you have the data at your fingertips.
AWS	ram	List all resources in this account that are shared with other accounts, or resources from other accounts that are shared with this account. Useful for cross-account attack paths.
AWS	role-trusts	Enumerates IAM role trust policies so you can look for overly permissive role trusts or find roles that trust a specific service.
AWS	route53	Enumerate all records from all route53 managed zones. Use this for application and service enumeration.
AWS	secrets	List secrets from SecretsManager and SSM. Look for interesting secrets in the list and then see who has access to them using use cloudfox iam-simulator and/or pmapper.
AWS	sns	This command enumerates all of the sns topics and gives you the commands to subscribe to a topic or send messages to a topic (if you have the permissions needed). This command only deals with topics, and not the SMS functionality. This command also attempts to summarize topic resource policies if they exist.
AWS	sqs	This command enumerates all of the sqs queues and gives you the commands to receive messages from a queue and send messages to a queue (if you have the permissions needed). This command also attempts to summarize queue resource policies if they exist.
AWS	tags	List all resources with tags, and all of the tags. This can be used similar to inventory as another method to identify what types of resources exist in an account.

Prowler

Prowler is an Open Source Security tool for AWS, Azure and GCP to perform Cloud Security best practices assessments, audits, incident response, compliance, continuous monitoring, hardening and forensics readiness. Includes CIS, NIST 800, NIST CSF, CISA, FedRAMP, PCI-DSS, GDPR, HIPAA, FFIEC, SOC2, GXP, Well-Architected Security, ENS and more. (Python)

Install

Pip package

Prowler is available as a project in PyPI, thus can be installed using pip with Python >= 3.9:

```
pip install prowler
prowler-v
```

More details at https://docs.prowler.cloud

Containers

The available versions of Prowler are the following:

- latest: in sync with master branch (bear in mind that it is not a stable version)
- <x.y.z> (release): you can find the releases here, those are stable releases.
- stable: this tag always point to the latest release.

The container images are available here:

- DockerHub
- AWS Public ECR

From Github

Python >= 3.9 is required with pip and poetry:

```
git clone https://github.com/prowler-cloud/prowler
cd prowler
poetry shell
poetry install
python prowler.py-v
```

Requirements

Prowler has been written in Python using the AWS SDK (Boto3), Azure SDK and GCP API Python Client.

AWS

Since Prowler uses AWS Credentials under the hood, you can follow any authentication method as described here. Make sure you have properly configured your AWS-CLI with a valid Access Key and Region or declare AWS variables properly (or instance profile/role):

```
aws configure
or

export AWS_ACCESS_KEY_ID="ASXXXXXXX"
export AWS_SECRET_ACCESS_KEY="XXXXXXXXX"
export AWS_SESSION_TOKEN="XXXXXXXXX"
```

Those credentials must be associated to a user or role with proper permissions to do all checks. To make sure, add the following AWS managed policies to the user or role being used:

- arn:aws:iam::aws:policy/SecurityAudit
- arn:aws:iam::aws:policy/job-function/ViewOnlyAccess

Moreover, some read-only additional permissions are needed for several checks, make sure you attach also the custom policy prowler-additions-policy.json to the role you are using.
If you want Prowler to send findings to AWS Security Hub, make sure you also attach the custom policy prowler-security-hub.json.

AWS

Use a custom AWS profile with -p/--profile and/or AWS regions which you want to audit with -f/--filter-region:

```
prowler aws --profile custom-profile -f us-east-1 eu-south-2
```

By default, prowler will scan all AWS regions.